Inteligencia ambiental en el Internet de las Cosas

Rubén Gómez Fuentes
Daniel Lago Aguado

Madrid, junio de 2016

Proyecto de la Facultad de Informática
Departamento de Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid

Trabajo Fin de Grado en Ingeniería del Software
Curso 2015-2016

Director: Gonzalo Pajares Martinsanz

La informática tiene que ver con los ordenadores lo mismo que la astronomía con los telescopios.

Edsger Wybe Dijkstra (1930-2002)

Autorización de difusión y utilización

Los abajo firmantes, matriculados en el Grado de Ingeniería del Software impartido por la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo de Fin de Grado, realizado durante el curso académico 2015-2016 y bajo la dirección de Gonzalo Pajares Martinsanz en el Departamento de Ingeniería del Software e Inteligencia Artificial, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet, y garantizar su preservación y acceso a largo plazo.

Rubén Gómez Fuentes **Daniel Lago Aguado**

Madrid, junio de 2016

Agradecimientos Personales

Antes de nada, queremos dedicar unas palabras de agradecimiento en término particular.

Rubén Gómez Fuentes

A mis padres, gracias por el inmenso sacrificio, apoyo, comprensión, cariño, y todo lo que aportáis hacia mi persona, después de un largo camino se ven los frutos que antaño se plantaron, sin vosotros nada de esto sería posible.

Hermanos, abuelos, resto de mi familia, gracias por creer y confiar en mí.

A mi pareja y su familia, gracias por todo el apoyo recibido, conseguís que todos los caminos recorridos y por recorrer sean más fáciles de afrontar.

A mis amigos, que como bien decía mi abuelo, los cuentas con los dedos de una mano, y con los años te das cuenta de la importancia de tenerlos cerca.

Mi compañero y amigo Daniel, gracias por haber compartido conmigo todas las batallas, para finalmente, poder realizar este trabajo juntos.

Gracias a todos los que, en algún momento de mi vida, han estado apoyándome, GRACIAS.

Daniel Lago Aguado

A mi familia y en especial a mis padres, quiero agradeceros el apoyo incondicional que me brindáis, la oportunidad de estudiar aquello que elegí y la confianza que habéis depositado en mí todos estos años.

A Gloria, gracias por aparecer en el momento apropiado y enseñarme que aún queda bondad en este mundo.

A mi amigo Rubén, te agradezco los momentos que hemos pasado juntos, no podría haber elegido mejor compañero.

A un pequeño monito, que se balancea en las lianas de mi memoria.

Agradecimientos

A nuestro director, Dr. Gonzalo Pajares, por dejarnos la libertad de elegir por nosotros mismos el proyecto que queríamos realizar, y depositar la confianza necesaria en nosotros para lograrlo.

Índice

Tabla de Figuras

Lista de Abreviaturas

IoT	*Internet of Things*
CSS	*Cascading Style Sheets*
CMS	*Content management system*
I2C	*Inter-Integrated Circuit*
GPIO	*General Purpose Input/Output*
HTML	*HyperText Markup Language*
AJAX	*Asynchronous JavaScript and XML*
JS	*JavaScript*
API	*Application Programming Interface*
MVC	*Model-View-Controller*
HTTP	*Hypertext Transfer Protocol*
IA	*Inteligencia Artificial*
RBPi	*Raspberry Pi*
JSON	*JavaScript Object Notation*
BBDD	*Base de Datos*
SQL	*Structured Query Language*
E/S	*Entrada / Salida*
TFG	*Trabajo Fin de Grado*
UCM	*Universidad Complutense de Madrid*

Resumen

En la actualidad, existe un concepto que está cobrando especial relevancia, el cual es conocido como IoT (Internet of Things, Internet de las Cosas) [1].

En el IoT [2] se define la interconexión digital de objetos cotidianos con internet, esto significa que no sólo “los humanos” tenemos la capacidad de conectarnos a internet, sino que caminamos hacia una nueva era donde prácticamente cualquier cosa podría ser conectada a internet, desde un reloj (smartwatch), como tenemos en la actualidad, hasta una nevera, una persiana, una sartén, etc.

En este proyecto se ha querido aplicar ciertas fases del IoT, para convertir una información ambiental poco sesgada, proporcionada por una pequeña estación meteorológica, en un valor adicional a la hora de tomar decisiones basadas en las variables ambientales, para determinar, según un proceso de aprendizaje automático, la sensación que una persona percibe en relación al tiempo meteorológico en un determinado momento. Para ello utilizamos una serie de sensores que se encargan de darnos la información ambiental necesaria (como la temperatura, humedad y presión atmosférica) una fuente de procesamiento como puede ser un micro-controlador, para después poder manejarla y procesarla en la nube, de forma remota, adquiriendo así el valor añadido que se espera en el IoT.

Además, en este proyecto se aplican técnicas de Inteligencia Artificial para ayudar al usuario en esa toma de decisiones, mediante un proceso de entrenamiento previo, que permite obtener información relevante para aplicarla posteriormente en el contexto meteorológico mencionado.

Para manejar todos estos conceptos y elementos, se hace uso de servicios Web, bases de datos, procesamiento y aprendizaje automático, integrando todos los servicios en una misma plataforma que facilite la comunicación de todos los elementos involucrados.

Palabras clave: Internet de las Cosas, IoT, Sensores, CMS, Inteligencia Artificial, Control Ambiental, Procesamiento en la nube, Aprendizaje automático, Aplicaciones Web, Transmisión de datos.

Abstract

Currently, there is a concept that is gaining special relevance, this concept is known as IoT (Internet of Things).

IoT defines digital interconnection of everyday objects to internet, this means humans will no longer be the only ones with the ability to connect to internet. We walk into a new era where anything could be virtually connected to Internet, from a clock (smartwatch), as we have today, to a refrigerator, a blind, a pan, etc.

In this project we have tried to apply certain phases of IoT, to convert a bit biased environmental information, in an additional value when making decisions about environmental conditions based on individual perceptions.
We use different sensors that are responsible for giving the necessary environmental information (such as temperature, humidity, or atmospheric pressure), a source of processing such as a micro-controller, and then appropriate abilities to handle and process the available information in the cloud, achieving the added value.

In addition, this project applies Artificial Intelligence techniques to assist the user in decision-making through preliminary training.

To handle all these things, we use Web services, databases, automated processing, integrating all those services on a single platform that facilitates communication of all these elements.

Keywords: IoT, Internet of Things, CMS, Artificial Intelligence, Environmental control, Cloud computing, Automatic learning, Web Applications, Data transmission.

1. Introducción

Este proyecto nace de la ambición e inquietudes de sus autores por explorar las nuevas tecnologías y sus posibilidades en un mundo en continua evolución.

El objetivo principal consiste en poder monitorizar una serie de datos que a priori pueden parecer poco relevantes, para que, mediante una serie de procesamientos, aplicando técnicas de Inteligencia Artificial, puedan suministrar al usuario datos de especial relevancia como ayuda a la toma de decisiones. Todo ello bajo el paradigma de lo que se conoce como Internet de las Cosas (IoT, Internet of Things), como concepto de vanguardia y en continuo auge, cada vez con mayor implantación en todos los órdenes de la vida, y especialmente en el mundo empresarial.

La orientación otorgada al proyecto se realiza pues, bajo el mencionado concepto IoT, para ello es necesario disponer de una serie de componentes y materiales, de bajo coste y bajo consumo para demostrar las amplias posibilidades de su uso en diversos ámbitos sin gran inversión, y a la vez posibilitar su ejecución sin financiación. Aquí se encuentran los sensores y la placa de procesamiento, así como las conexiones entre los mismos, con la perspectiva de que simultáneamente permitieran una alta flexibilidad y escalabilidad para su adaptabilidad al mayor número de situaciones.

La adquisición de todos los componentes hardware necesarios para proceder a la captura de los datos y su posterior obtención, constituyó la primera fase del IoT.

Posteriormente, mediante la técnica de *scripts* se procedió a la transmisión de estos datos hacia diferentes aplicaciones o servicios, siendo ésta la segunda fase de IoT.

La tercera fase del IoT hace referencia al análisis de los datos capturados, que es un proceso orientado a inspeccionar, limpiar y transformar dichos datos con el objetivo de extraer información útil, lo que sugiere conclusiones, y apoyo a la toma de decisiones. En esta fase es donde se incluyen técnicas de Inteligencia Artificial, más específicamente de aprendizaje automático, para el propósito indicado. Para facilitar esta tarea se han integrado los datos en un gestor de contenidos CMS [3] [4] Open Source, concretamente emonCMS [5].

Introduction

This project arises from the ambition and concerns from authors to explore new technologies and their possibilities in a changing world.

The main objective is to monitor a series of data that may seem irrelevant at a first glance, however through different processing methods and applying artificial intelligence techniques, they can provide the user information of particular relevance as an aid for decision making. All under the paradigm of what is known as the Internet of Things (IoT), as a steadily growing avant-garde concept, with an increasing presence in all life's layers, and especially in business's world.

This project is conducted by the aforementioned IoT concept. It is necessary to have a number of components and materials, low cost and low power consumption, in order to show the wide possibilities of use in several areas without a large investment, being able to execute it without funding. Here are the sensors and processing board and the connections between them, with the objective of simultaneously to allow high flexibility and scalability, leading to a better adaptability.

The acquisition of all required and necessary hardware components to proceed with the data acquisition, make up the first phase of IoT.

Subsequently, using the technique of scripts we proceeded to transmit these data to different applications or services, which is the second phase of IoT.

The third phase of IoT refers to the analysis of the captured data, which is designed to inspect, clean and transform the data in order to extract useful information, suggesting conclusions, and supporting decision-making. This phase is where Artificial Intelligence techniques, more specifically machine learning, are included. To facilitate this task we have decided to integrate this data in a content management system CMS [3] [4] Open Source, specifically emonCMS [5] defining an structrure with the possibility of applying cloud computing techniques

1.1. Objetivos y plan de trabajo

El planteamiento que sustenta el proyecto, consiste en simular un problema real como puede ser el siguiente: una empresa de domótica solicita el diseño de un sistema que permita ayudar a sus clientes en la toma de decisión del vestuario, en base a parámetros atmosféricos, tales como la temperatura, humedad o presión, y que éste sea perfectamente personalizable en el sentido de que cada persona, usuaria del sistema, pueda tomar una decisión concreta y subjetiva en función del análisis de datos en conjunción con su propia decisión. Esto es, bajo las mismas condiciones meteorológicas, distintas personas pueden valorar su interpretación de distinta forma y elegir en consecuencia un vestuario diferente para salir a la calle.

Tras las preceptivas reuniones con el cliente, se elabora el correspondiente documento de Especificación de Requisitos Software (ERS) donde se detalla la visión general del producto, funcionalidad del sistema a diseñar y requerimientos necesarios.
Técnicamente, la visión general del producto consiste en el desarrollo de una aplicación informática que permita al cliente de esta empresa la visualización de datos atmosféricos que el cliente considere relevantes, a través de una plataforma web, fácilmente accesible, y prioritariamente proporcionando el soporte necesario como ayuda a la toma de decisiones basado en técnicas de Inteligencia Artificial [6] [7], más concretamente mediante aprendizaje automático, que incluyen entrenamiento previo por parte del usuario, con el fin de establecer una predicción de su sensación térmica individualizada para cada usuario y en cada momento.

En consecuencia, se plantean los siguientes requisitos:

1. Distinguir entre la monitorización visual de los datos y el módulo de Inteligencia Artificial.
2. En la monitorización de los datos se visualizan tablas históricas donde el usuario puede establecer diferentes intervalos de tiempo, tales como, semanas, meses o años.
3. El módulo de IA debe estar estructurado de forma que permita la implementación de diferentes algoritmos de aprendizaje, haciéndolo fácilmente escalable.
4. El sistema no debe limitar el número de sensores desde los que se captura los datos, es decir, que dicho número debe ser escalable.
5. La interfaz de usuario para el módulo de IA debe ser amigable, simple y funcional, para garantizar y fomentar su usabilidad.

1.2. Estructura de la memoria

La memoria está distribuida en seis capítulos, claramente diferenciados, en los que se detalla todo el proceso de investigación y desarrollo del mismo. Además, se incluyen cuatro apéndices con el fin de matizar y dejar suficientemente claros algunos aspectos de la aplicación y su funcionamiento.

El capítulo 1, se compone de una breve introducción sobre la razón de ser de este proyecto, la estructuración de la memoria y a las contribuciones personales de cada miembro al proyecto.

En el capítulo 2, se detallan todos los componentes necesarios para la realización del proyecto, tanto elementos hardware como elementos software.

El capítulo 3, es el más extenso, ya que se detallan todas las fases de las que se compone el proyecto, estas fases constituyen el núcleo principal de la memoria. En el planteamiento y diseño que se formula, se distinguen tres fases fundamentales: captura, análisis y transmisión de los datos, dichas fases siguen la estructura general de una aplicación orientada hacia el IoT. Por otra parte, las etapas principales del desarrollo del proyecto tales como la captura de datos, el procesamiento y la lógica de la aplicación y por último la visualización e interacción, están recogidas en cada una de las fases del IoT. Además, cada una de estas fases se aborda considerando, el estado del arte, el diseño y la implementación específica. El hecho de tener un estado del arte por cada módulo de desarrollo del proyecto, responde a la comodidad que ofrece al lector el hecho de disponer en cada momento la información previa requerida para entender el desarrollo según se va necesitando.

En el capítulo 4, se exponen los resultados y consideraciones relevantes derivadas del proyecto.

En el capítulo 5, se incluyen las principales conclusiones del trabajo realizado. También se muestran posibles mejoras que se pueden incorporar en futuras versiones de desarrollo de la aplicación o cómo orientarlas hacia diferentes aplicaciones o usos.

La memoria contiene 4 apéndices para detallar aspectos de interés y útiles, tales como código del programa, manual de usuario, manual de instalación y todos aquellos datos que sean relevantes para el proyecto, pero que se separan del cuerpo principal del documento a fin de no sobrecargarlo.

1.3. Contribuciones personales

1.3.1. Rubén Gómez Fuentes

Contribuciones al proyecto:

Desde hace unos años atrás, tanto mi compañero como yo teníamos varios proyectos que nos gustaría poder realizar, uno de ellos, idea de mi compañero Daniel, era un “ordenador de a bordo” (utilizando una Raspberry Pi / arduino, varios sensores de proximidad, antena GPS, display, etc.), se presentó en una de las asignaturas de Ingeniería del Software, pero por votación entre los integrantes de la clase no se llegó a realizar. Otro de los proyectos, en este caso, idea mía, era una “Estación Meteorológica”, usando también una Raspberry Pi/ arduino, sensores de humedad, temperatura, y presión atmosférica.

Se tomó la decisión de realizar como trabajo fin de grado este último por varios motivos. El primero, que el coste de asumir todos los componentes era menor. El segundo, que no requería ningún tipo de instalación física compleja tal como pudiera ser en un coche por ejemplo (como sí que lo requiere el ordenador de a bordo) y por último, y más importante, nos daba más libertad a la hora de aplicar mecanismos de inteligencia artificial.

Al principio esta idea de una "Estación Meteorológica" me surgió al querer intentar controlar toda la monitorización de temperatura y humedad en casa, para comprobar el rendimiento de un nuevo sistema de calefacción que tenía instalado en el piso, además ya había comprado previamente una Raspberry Pi que la usaba para otro tipo de finalidad y me gustaría poder darle este nuevo uso.

Una vez decidido el proyecto y aprobado por nuestro tutor hubo que comprar los materiales, las Raspberry Pi ya las teníamos, entonces procedí a hacer un trabajo de investigación para ver qué sensores eran los más convenientes para nuestro proyecto, y que, a su vez, estuvieran orientados de una manera clara al IoT, es decir, que fueran de bajo consumo y bajo coste, con facilidad de comunicación mediante internet.

Después de algunas semanas investigando, decidí que los sensores que mejor se adaptaban a nuestro proyecto eran: El BMP 1080 de bosch y el DHT22.

Llegué a esta conclusión porque el sensor BMP1080 está integrado en una placa con interfaz I2C que nos facilitaba la comunicación entre el sensor y la Raspberry Pi, este sensor nos ofrecía datos de temperatura y presión atmosférica.

Faltaba un dato crucial, la humedad, entonces opté por un sensor de bajo coste muy habitual como puede ser el DHT11 o su versión mejorada, DHT22, que nos ofrece humedad y también temperatura. Este sensor carece de interfaz I2C y se conecta directamente a uno de los pines de datos GPIO de la Raspberry Pi; además, a este sensor hubo que añadirle una resistencia de 4,7k.

Además de los sensores, que son elementos hardware fundamentales del proyecto, necesitábamos otros elementos para poder conectar de una forma fácil y estable los sensores.

La Raspberry Pi cuenta con una interfaz de pines llamada GPIO (General Purpose Input/Output) es, como su propio nombre indica, un sistema de E/S por el cual íbamos a realizar la comunicación entre los sensores y la RBPi, para facilitar esta tarea creí conveniente comprar una placa Protoboard donde irán “pinchados” los sensores, y un GPIO Extender, que servirá de puente entre los fines GPIO de la RBPi y la protoboard.

Fue necesario también soldar el BMP1080 a una serie de pines para fijarlos en la protoboard, acción realizada con éxito.

Una vez bien definida toda la estructura hardware, procedí a la búsqueda de una serie de drivers para completar la comunicación entre los sensores y la RBPi, para poder enviarle peticiones y que éstos respondieran debidamente, esta tarea resultó más compleja de lo esperado, teniendo que buscar toda una serie de librerías software específicas para cada uno de los sensores.

Resuelta la sincronización entre los diferentes componentes, pensamos en montar un servicio web para poder monitorizar esos datos desde la nube. En un principio, decidí integrarlo en un servicio web ya existente llamado RPi-Monitor.

RPI-Monitor es un servicio web que se encarga de monitorizar datos relevantes de la RBPi como pueden ser: consumo de memoria RAM, carga de CPU, temperatura de la CPU en tiempo real, etc.

La integración de los datos en esta aplicación nos permitiría comprobar fácilmente el correcto funcionamiento del muestreo de datos hacia la nube, y mostrarlos cómodamente en una serie de tablas predefinidas.

Una vez integrado, se planteó la realización de un módulo de Inteligencia Artificial que fuera capaz de predecir una sensación térmica personalizada en cada momento, con RPI-Monitor esto no era viable, teniendo en este momento dos opciones, o crear una aplicación nueva, partiendo desde la base, o cambiar de servicio. Después de varias semanas encontré una herramienta Open Source que encajaba bastante bien con lo que queríamos hacer y nos permitía escalar y crear nuestro módulo sin demasiados problemas.

Esta herramienta se llama EmonCMS, como su nombre indica, se trata de un CMS (Content Management System), sistema de gestión de contenidos que nos permitiría, por un lado, mostrar todos los datos recogidos por los sensores (calcular otros ligados a los que ya nos proporciona el sensor, por ejemplo, la altitud, que es un valor calculado). Y, por otro lado, nos permitiría crear el módulo de Inteligencia Artificial deseado. Finalmente se tomó la decisión de usar esta herramienta con todas sus consecuencias.

Utilizamos la API de emonCMS, estudiamos su funcionamiento y empezamos a trabajar con ello.

Por otro lado, dentro del emonCMS y más concretamente en el módulo de Inteligencia Artificial, me encargué de toda la parte referente a la interfaz gráfica de cara al usuario, utilizando tecnologías CSS, HTML, JQuery y JavaScript. También fue de mi responsabilidad la creación de un dashboard propio para la monitorización de todos los datos.

En cuanto a la realización de la memoria, al haber estado mi parte bastante enfocada a la investigación de todos los componentes y herramientas utilizadas, me he encargado principalmente del estado del arte de cada una de las fases, resumen e introducción al proyecto, así como la descripción de todos los componentes hardware y software utilizados. También en la construcción de todos los apéndices, donde se muestra con detalle los diferentes procesos que se han llevado a cabo para la instalación, despliegue y utilización de la aplicación. Por otro lado, y no menos relevante, toda la maquetación de la memoria.

1.3.2. Daniel Lago Aguado

En un principio no teníamos claro qué queríamos hacer como trabajo de fin de grado, tanto mi compañero como yo propusimos distintas ideas. Recuerdo que de todas esas ideas llevamos a discutir con nuestro director dos de ellas, mi compañero propuso desarrollar una estación meteorológica y yo propuse el desarrollo de una impresora que realizara grabados mediante un diodo láser. Mi propuesta resultó ser demasiado compleja por lo que nos decantamos por la estación meteorológica. Hicimos una investigación previa sobre cómo funciona una estación y resultó que ésta requería muchos más componentes de los que habíamos planteado. Cosas como un pluviómetro, una veleta, etc. Adquirir todos estos componentes superaba el presupuesto disponible, de manera que reestructuramos la idea. Propuse plantear el trabajo de forma que desarrollásemos un prototipo de sistema que en un futuro pudiese comercializarse. Algo que una persona pudiera comprar e instalar de forma simple en su casa, sin llegar a desviarnos demasiado del planteamiento original. Nuestro sistema debía ofrecer al usuario información detallada sobre condiciones ambientales que pudieran resultarle de interés con la mínima dificultad posible en la instalación y el uso.

Tanto mi compañero como yo disponíamos de una Raspberry Pi cada uno, sin embargo, yo también tenía un arduino. Consideré las ventajas de cada uno. El arduino consume mucha menos energía, nos facilita la conexión de los sensores con el sistema, tiene entradas analógicas que nos permitirían comprar sensores más baratos y además yo ya tenía experiencia con este microcontrolador. Pero teniendo en cuenta que con el fin de comercializarlo queríamos tener tanto la captura, como el almacenaje y la muestra de datos en un mismo dispositivo, me acabé decantando por hacer uso de la Raspberry Pi. Una vez decidido esto busqué información sobre los GPIO y qué tipo de sensores íbamos a necesitar. El sensor de temperatura que tenía para mi arduino aparentemente no servía, ya que la Raspberry Pi no dispone de entradas analógicas. Le pedí a mi compañero que buscara y comprara un sensor de humedad y otro de temperatura. Debido a la distribución de los pines GPIO busqué un extensor que nos facilitara el trabajo en la instalación y manipulación de los circuitos. Encontré un componente llamado “cobbler”, que permitía conectar directamente todos los pines a una placa de prototipado. Tanto los sensores como el cobbler llegaron más o menos en la misma fecha, así que tras tener todos los componentes necesarios procedí a instalarlos. El BMP180 (temperatura y presión) fue significativamente más caro, pero sin ninguna duda el que mejor calidad demostró tener. Venía ya ensamblado con la resistencia necesaria y el circuito preparado para una interfaz i2c.

Este sensor disponía de un conjunto de pines para que tras ser soldados permitiesen la conexión con la placa de prototipado. Para las pruebas iniciales enrollé un poco los cables de forma que hicieran contacto con la placa. Después conecté el DHT22 con su respectiva resistencia de 4,7 kilo ohmnios. Investigué la forma de leer los datos. Lo más sencillo parecía ser usar una librería de Python que ofrecía el propio vendedor. Para comprobar que la conexión con el driver era correcta seguí una serie de pasos que encontré en un tutorial donde hacían uso de un conjunto de herramientas llamado i2ctools. Las pruebas resultaron correctas así que hice uso de las librerías de adafruit para los dos sensores. En un principio los dos sensores parecían funcionar a la perfección, sin embargo, el DHT22 en algunas de las ejecuciones no devolvía ninguna lectura. En ese momento no se consideró importante y continuamos con el siguiente paso. Para la visualización de datos mi compañero encontró una aplicación web llamada rpi-monitor que permitía manipular y mostrar algunos valores en la Raspberry Pi. La funcionalidad de esa aplicación no se ajustaba demasiado a la que nosotros queríamos sin

embargo tenía un sistema de visualización que resultaba visualmente atractivo, por lo que decidí explorarlo más a fondo. Estudié el funcionamiento de esta aplicación con el objetivo de utilizarla en nuestro proyecto, pero con el paso del tiempo se reforzaba mi opinión inicial, esta aplicación no encajaba con nuestros objetivos. A raíz del sistema de monitorización de energía mi compañero me propuso usar el cms que ofrecía esta compañía. El siguiente paso fue leerme toda la documentación que ofrecía emonCMS. No había punto de comparación respecto a rpi-monitor. La documentación era fácilmente accesible, extensa, descriptiva, clara. Desde mi humilde punto de vista este cms es un buen ejemplo lo que es la ingeniería aplicada al software. Tenía muchos de los factores de calidad que se exigen al software. Siguiendo los pasos que se muestran en su github sobre cómo enviar datos al cms, modifiqué los drivers para que directamente realizaran el envío. Una vez conseguí que se visualizaran los datos que transmitía el sensor estuvimos probando las diferentes opciones que ofrecía el cms. Los elementos de visualización se conocen como widgets y son una representación de uno o varios datos que se reciben como alimentación. Diseñamos una composición de widgets básica, reuniendo los datos principales a mostrar. Mi compañero incluyó algún dato más como la presión atmosférica o la altitud.

Ya teníamos una visualización de los datos bastante atractiva, pero sentía que le seguía faltando algo a la aplicación. Por esta razón propuse llevar a cabo un análisis de estos datos mediante algún algoritmo de inteligencia artificial, aprovechando los conocimientos adquiridos en la asignatura de ingeniería del conocimiento. Planteé que podíamos utilizar estos datos para que el sistema decidiera en función de un entrenamiento por parte del usuario cual iba a ser la sensación térmica que iba tener esta persona. La mayor dificultad de esta parte del proyecto era desarrollar una estructura que permitiese tener distintos algoritmos integrados con el cms. Tomé como ejemplo la implementación de los dashboard, que a mi juicio se parecía mucho a la capa intermedia que queríamos implementar. Un dashboard, en este contexto, te permite tener distintos conjuntos de widgets. Mi objetivo era tener un dashboard que nos permitiese almacenar un tipo de algoritmo de IA en él, de forma que pudieses tener tantos dashboard como necesites y del tipo de algoritmo que quieras. Este concepto lo bauticé como IAentity, ya que realmente representaba una entidad en la base de datos. Tras muchos días de lectura y compresión del código del dashboard y de la estructura de emonCMS, la IAentity estaba completamente fusionada con el cms. Para la implementación del algoritmo de IA consideré que el algoritmo que más se ajustaba a nuestro problema era el clasificador de bayes. Los ejemplos vistos en clase hacían uso de coordenadas en el espacio, por lo que a mi juicio encajaba muy bien con los valores de temperatura y humedad, pudiendo convertirse estos a su vez en coordenadas. Me preocupaba que al haber una mayor amplitud de rango en la humedad que en la temperatura esto pudiera producir un margen de error amplio en la predicción de la sensación térmica. Sin embargo, tras implementarlo y entrenarlo con varios casos de pruebas, demostró ajustarse muy bien a los requisitos que se le exigían.

En cuanto a la redacción de este documento, propuse la estructura que hemos seguido finalmente. Me centré en las fases del proyecto, introduciendo los conceptos previos, realizando los diagramas, detallando la implementación, etc. También introduje las conclusiones y el abstract. Me gustaría mencionar que ha sido complejo el hecho de distinguir cuales han sido las aportaciones de cada miembro. A diferencia de lo que pasa con un grupo más grande, en este proyecto no ha habido una separación estricta de tareas. Todas las ideas, todas las propuestas, incluso en la implementación de código, han sido comentadas, supervisadas y acordadas por ambos miembros del grupo. Considero que el trabajo en equipo ha sido un factor clave en el desarrollo de este proyecto.

2. Descripción de los componentes

2.1. Introducción

Como principio fundamental del IoT [2], nuestro proyecto debe estar formado por componentes de bajo consumo, por lo tanto, fue uno de los criterios principales aplicados para la adquisición de los componentes hardware finalmente seleccionados.

Otro criterio utilizado es el de simplicidad de uso. Para ello se optó por elegir componentes que simplificaran al máximo el trabajo a la hora de implementar las funcionalidades especificadas y definidas en los requisitos del proyecto.

Además de los componentes hardware requeridos, es necesario proporcionar la correspondiente estructura software para dar soporte e integrar todos los componentes físicos y lógicos involucrados en la aplicación.

2.2. Componentes Hardware.

2.2.1. Raspberry Pi 2

Raspberry Pi [8] es un ordenador de placa reducida. El diseño se centra en un sólo microprocesador con su memoria RAM, E/S y todas las demás características de un computador funcional en una sola tarjeta que, como se indica previamente, suele ser de tamaño reducido, y que tiene todo lo que necesita en la placa base, siendo además de bajo coste, desarrollado en Reino Unido por la Fundación Raspberry Pi [5], con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas. La figura 1 muestra una vista general de este componente.

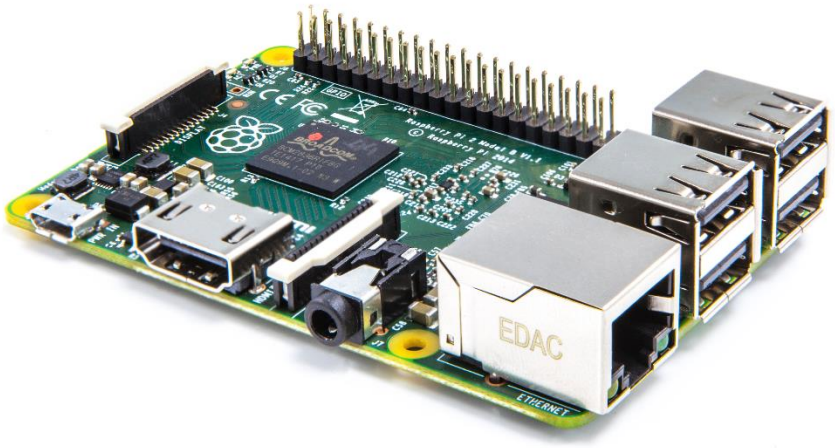


Figura 1: Raspberry Pi

Se ha elegido la Raspberry Pi frente a otros competidores como podría ser arduino, debido a que aquella ofrece toda la funcionalidad de un ordenador convencional, con lo que esto conlleva. Además, permite el uso de cualquier lenguaje de programación, facilidad de comunicación o interconexión de los componentes.

La RBPi utiliza un conjunto de pines conocidos como GPIOS [9] [10], los cuales permiten la comunicación del dispositivo con distintos componentes electrónicos. Además, incorpora un bus de datos serial conocido como interfaz I2C [11].

2.2.2. BMP1080

El sistema incluye un sensor de presión barométrica identificado como BMP180 [12] de alta precisión. Este sensor también proporciona datos de temperatura.

Sus características y especificaciones técnicas principales son las siguientes:

- Rango de medición de presión: de 300 a 1100 hPa.
- Margen de error mínimo: 0.03 hPa.
- Rango de medición de temperatura: de -40°C a 85 °C.
- Precisión de medición de temperatura: <±0.5 °C.
- Alimentación: de 1.8V a 3,6 Vdc.

Está diseñado para ser conectado directamente a un microcontrolador mediante su interfaz I2C. Dispone de dos resistencias integradas en la propia placa de 4,7k.

La figura 2, muestra exactamente dicho sensor junto con sus componentes e interconexiones.

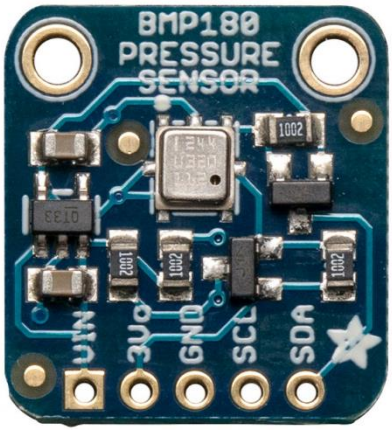


Figura 2: Sensor BMP 180

2.2.3. DHT22

El DHT22 [13] es un sensor digital de temperatura y humedad, de bajo coste. Utiliza un dispositivo capacitivo de humedad, así como un termistor para medir el aire circundante.

Las especificaciones técnicas principales de este sensor son las siguientes:

- Rango de medición de humedad: de 0 a 100% RH.
- Precisión de medición de humedad: 2% RH
- Rango de medición de temperatura: -40°C a 125 °C.
- Precisión de medición de temperatura: <±0.5 °C.
- Alimentación: de 3.3Vdc a 6Vdc.

El único inconveniente de este sensor, es que sólo proporciona datos con una frecuencia de refresco de 2 segundos.

Es necesario también, conectarle una resistencia de 4,7k para su correcto funcionamiento.

La figura 3, muestra un ejemplo de este elemento.

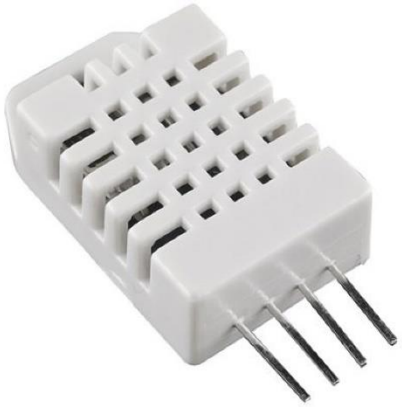


Figura 3: Sensor DHT22

2.2.4. Protoboard.

Una protoboard [14] o placa de pruebas, consiste en un tablero con orificios que se encuentran conectados eléctricamente entre sí de manera interna, habitualmente siguiendo patrones de líneas, en el cual se pueden insertar componentes electrónicos y cables para la interconexión de circuitos electrónicos y sistemas similares.

Normalmente está hecho de dos materiales, un aislante, generalmente un plástico, y un conductor que conecta los diversos orificios entre sí.
La figura 4, muestra este componente.

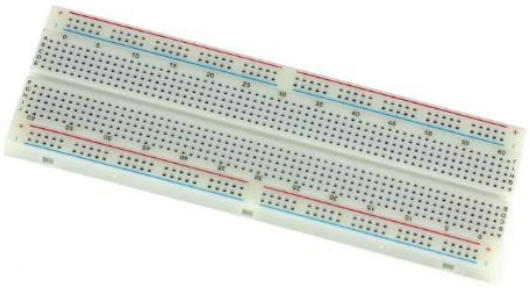


Figura 4 : Placa Protoboard

2.2.5. GPIOs Extender.

Este conector de extensión para los puertos GPIO de Raspberry Pi, permite “extender” los puertos GPIO de la RBPi a la placa protoboard, para así poder fijar los sensores en un mismo lugar y facilitar las conexiones entre los mismos.

La figura 5, muestra un ejemplo de este componente.

Figura 5: Extender GPIOs

2.2.6. Otros componentes.

Otros materiales utilizados para el desarrollo del proyecto son:

- Cables para las conexiones.
- Soldador de estaño, y estaño, para soldar el sensor BMP180.
- Multímetro.
- Alimentación para la Raspberry Pi de 1A.

2.3. Componentes Software

2.3.1. CMS.

CMS [3] [4] son las siglas de Content Management System, que se traduce directamente al español como Sistema Gestor de Contenidos. Como su propio nombre indica, es un sistema que permite gestionar contenidos, tratándose realmente de una aplicación informática usada para crear, editar, gestionar y publicar contenido digital multimedia. El gestor de contenidos genera páginas web dinámicas interactuando con el servidor web para generar la página web bajo petición del usuario, con el formato predefinido y el contenido extraído de la base de datos del servidor.

2.3.1.1. RPi-Monitor.

RPi-Monitor [15] es una aplicación Web que permite monitorizar ciertos parámetros de la propia Raspberry Pi, tal como la carga de la CPU, el consumo de memoria RAM y la temperatura del procesador en tiempo real. También permite generar estadísticas y visualizar la evolución y el comportamiento de estos parámetros a lo largo de un período de tiempo determinado. Es esta aplicación, la que, en una primera instancia, nos ha permitido integrar los datos recogidos por los sensores, y a la vez realizar una representación gráfica y dinámica en el tiempo.

La misma aplicación se encarga de invocar al comando establecido al efecto, utilizar su salida para incorporar los datos actualizados a las gráficas, que se utilizan para la monitorización y visualización de los datos recibidos desde los sensores.

En realidad, RPi-Monitor no es un CMS como tal, sino una aplicación web, y carece de la flexibilidad que tiene un CMS convencional. Por esta razón, pese a una primera implementación en esta herramienta, fue descartada en beneficio de un sistema más flexible y escalable. En la figura 6, se muestra el logo representativo propio de RPi-Monitor.

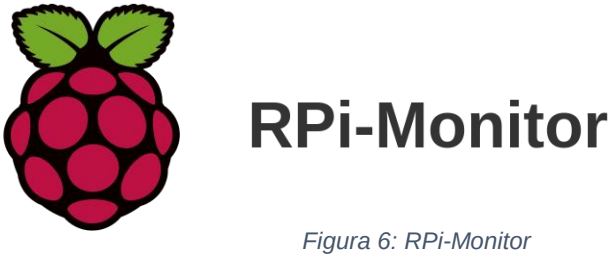


Figura 6: RPi-Monitor

2.3.1.2. EmonCMS.

EmonCMS [5] es un CMS de software libre desarrollado por la empresa OpenEnergyMonitor [16]. Es una aplicación diseñada para el procesamiento, registro, y visualización de energía, temperatura y otros datos de carácter ambiental.
Por consiguiente, esta herramienta se adapta perfectamente a las necesidades del proyecto que se plantea, permitiendo la captura de datos de una forma completamente modular y escalable, a la vez que proporciona una interfaz gráfica adecuada para el tipo de información que se pretende monitorizar.
Por este motivo, ha sido este CMS el componente software finalmente seleccionado para poder desarrollar íntegramente el proyecto propuesto. En nuestro caso, hemos instalado este CMS en la Raspberry Pi.
En la figura 7, se muestra el logo representativo de emoncms.

Figura 7: EmonCMS

3. Desarrollo del proyecto

Esta sección describe el desarrollo del proyecto siguiendo la división de fases propias establecidas en el ámbito del IoT. Cada fase consta de diferentes apartados, como son: estado del arte, diseño e implementación.
En el estado del arte, se definen una serie de conceptos previos que deben establecerse como paso previo a las fases siguientes.
En el diseño, se realiza una esquematización para facilitar el proceso de desarrollo de la aplicación, a la vez que sirve para entender el funcionamiento de la misma.
En la implementación, se describen los pasos seguidos en el desarrollo de la fase del IoT correspondiente.

3.1. Captura de datos

3.1.1. Estado del Arte

Un sensor, es un dispositivo capaz de detectar magnitudes físicas o químicas, llamadas variables de instrumentación, y transformarlas en variables eléctricas. Las variables de instrumentación pueden ser, por ejemplo: intensidad luminosa, temperatura, distancia, aceleración, inclinación, presión, desplazamiento, fuerza, humedad, movimiento etc.

Los sensores se conectan a la Raspberry Pi con unos pines conocidos como GPIO, tal y como se ha indicado previamente, los cuales se encuentran distribuidos de la siguiente forma que aparece en la figura 8

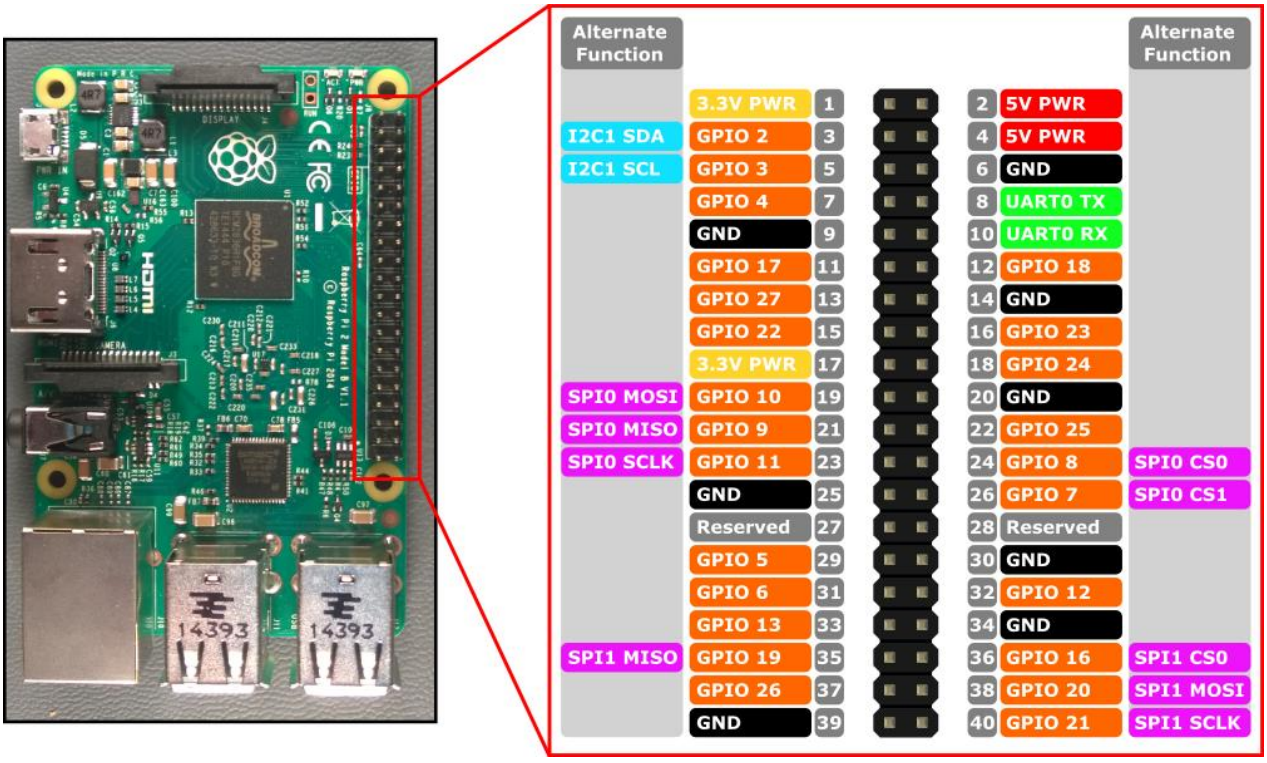


Figura 8: Esquema GPIOs Raspberry Pi 2

Debido a que la RBPi no posee entradas analógicas, se dispone de una interfaz conocida como interfaz i2c que permite transmitir la lectura del sensor al dispositivo.

El I2C (Inter-Integrated Circuit) [11] [17] está diseñado como un bus maestro-esclavo. La transferencia de datos es siempre inicializada por un maestro, mientras que el esclavo reacciona. Es posible tener varios maestros (Multimaster-Mode).

En el modo multimaestro, se pueden comunicar dos maestros entre sí, de modo que uno de ellos trabaja realmente como esclavo.

El arbitraje, (control de acceso en el bus) se rige por las especificaciones propias del interfaz I2C, de este modo los maestros pueden ir turnándose entre ellos para asumir el rol de maestro según diferentes instantes de tiempo.

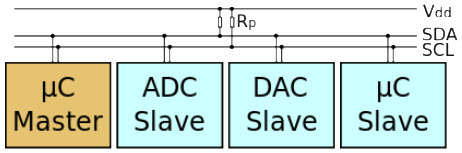


Figura 9: I2C

3.1.2. Diseño

En esta subsección, se incluyen las abstracciones necesarias relativas al funcionamiento de la aplicación a nivel de la captura de datos.

3.1.2.1. Diseño general

El controlador del dispositivo o driver específico del sensor, se comunicará con éste, solicitándole que devuelva la lectura de datos a través de la interfaz i2c. Esta lectura se realizará en intervalos de dos segundos, siguiendo especificaciones del cliente. Posteriormente el driver envía estos datos al CMS para su tratamiento y procesamiento.

En la figura 10, se muestra un diagrama que describe la comunicación entre los distintos componentes del sistema.

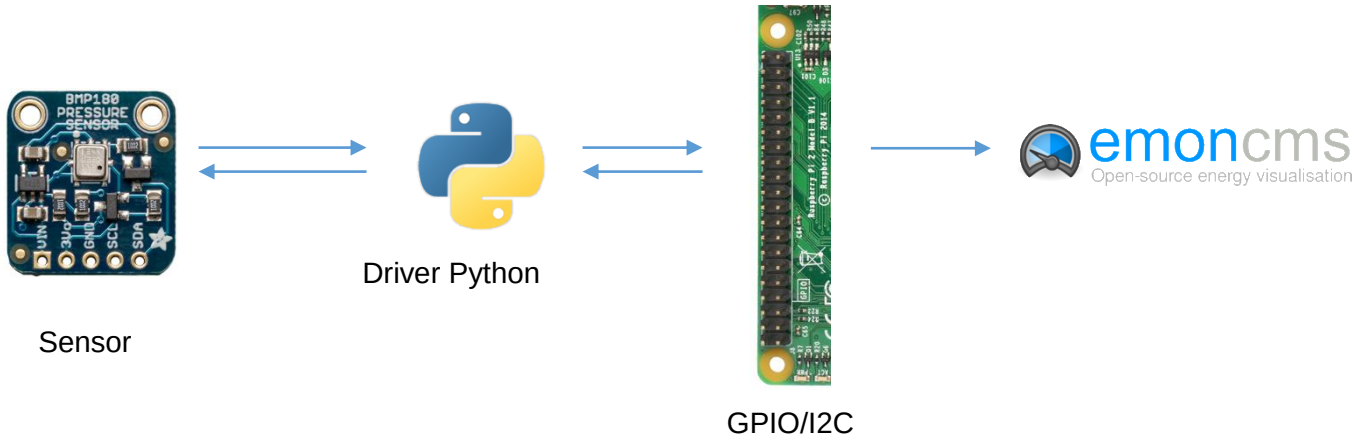


Figura 10: Esquema general de diseño

3.1.2.2. Diseño de conexión

La figura 11, muestra gráficamente el diagrama de la conexión física entre los sensores y los GPIO de la Raspberry Pi.

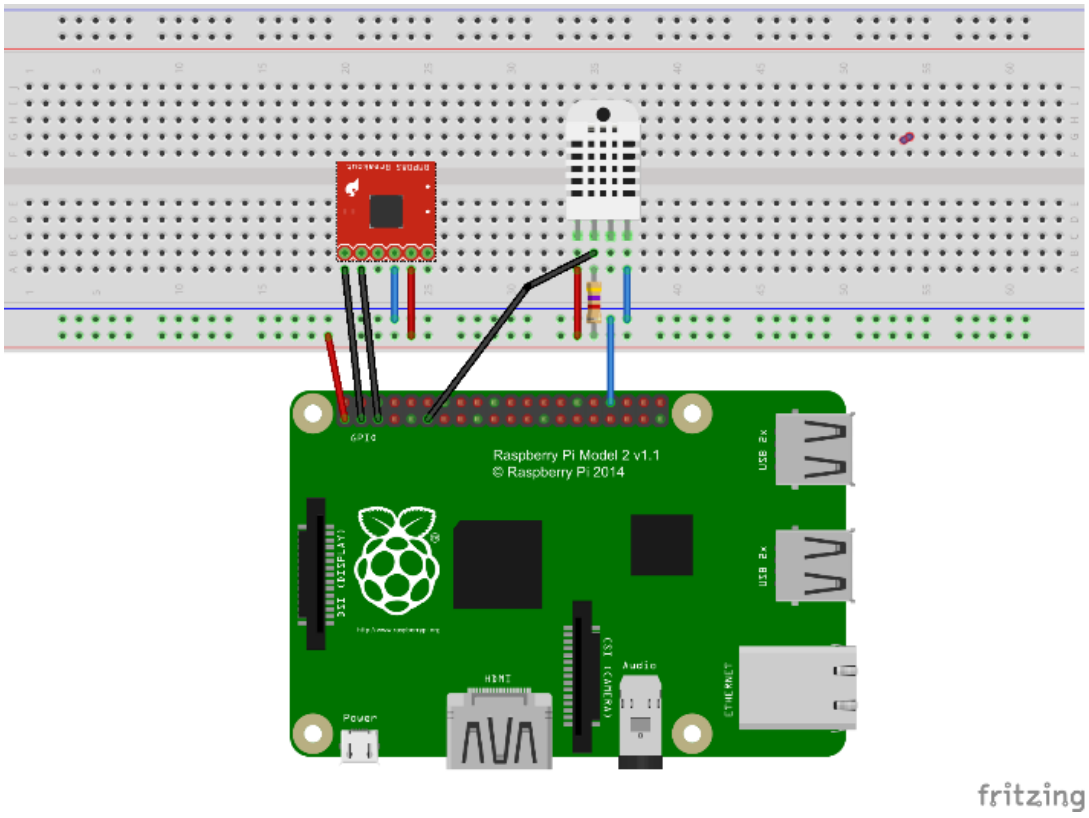


Figura 11: Esquema general de conexión de sensores con Raspberry Pi 2

En la figura 12, se describe de forma gráfica la conexión desde la perspectiva de un circuito electrónico.

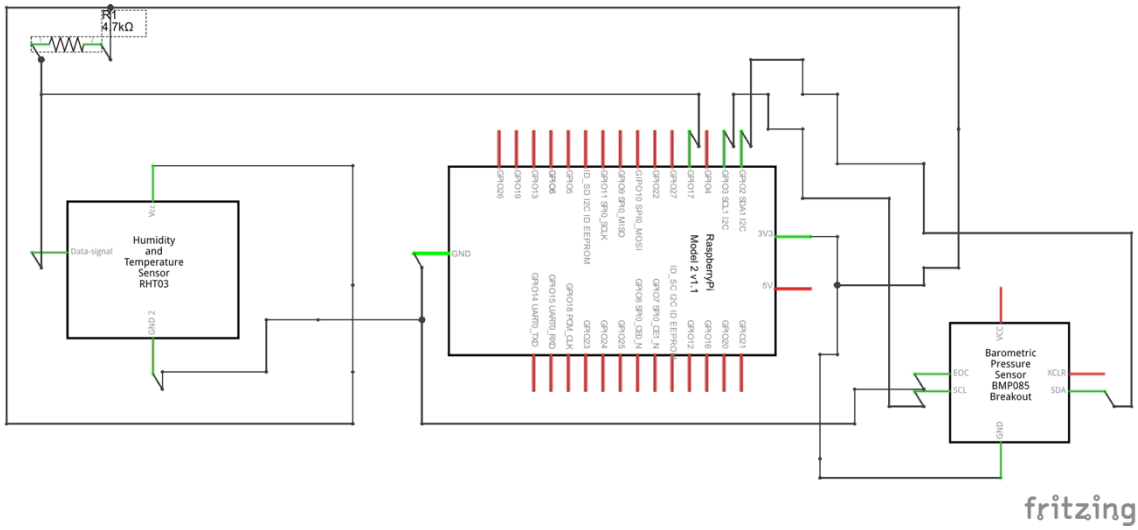


Figura 12: Esquema general de conexión de sensores con Raspberry Pi 2 - 2

3.1.3. Implementación

3.1.3.1. Interfaz I2C

I²C (Inter-Integrated Circuit) [11], es un bus de datos serial desarrollado en 1982 por Philips Semiconductors. Este bus se basa en tres señales:

- SDA, (System Data) que permite la transmisión de los datos entre los dispositivos.
- SCL, (System Clock) por donde se propagan los pulsos de reloj que sincronizan el sistema.
- GND, (Ground) Interconectada entre todos los dispositivos conectados al bus.

Por defecto, la interfaz I2C no viene habilitada en la Raspberry Pi, por lo tanto, la primera operación a realizar es su habilitación, a la vez que se instalan todas las herramientas necesarias para su utilización, como, por ejemplo:

- i2c-tools: utilidades de consola para controlar el bus.
- i2cdetect: Para determinar los dispositivos que están conectado.
- python-smbus: Herramientas para poder utilizar el bus con Python.

En el apéndice A se explica detalladamente el proceso de instalación de estos componentes.

3.1.3.2. Lenguaje de programación

Python [18], es el lenguaje de programación recomendado por los fundadores del proyecto Raspberry Pi, dado que es un lenguaje con una sintaxis sencilla y clara, es recomendado en los ámbitos educativos.

Además de ser un lenguaje interpretado o de script, fuertemente tipado y dinámico, es multiplataforma y orientado a objetos.

En este sentido, python es un lenguaje con potencialidad suficiente y con posibilidad de integrar diversas librerías que facilitan la realización de diversas tareas y operaciones. Por todo ello, se ha decidido utilizar las librerías en python que ya proporciona Adafruit [19] para el tipo de sensores adquiridos y utilizados en el desarrollo del proyecto.

Para la implementación de las funcionalidades de los sensores ha sido necesario el uso de librerías como las siguientes:

- Adafruit_BMP_Driver [20]
- Adafruit_DHT_Driver [21]

- Adafruit Python GPIO Library - (Adafruit_I2C) [22]

Estas librerías, publicadas por Adafruit, facilitan enormemente las cosas a la hora de capturar la información de los sensores, ya que, en ellas, están implementadas todas las funciones necesarias para interactuar con los diferentes sensores a través de la interfaz I2C.

Una vez instaladas las librerías, se crean los propios scripts en python (se incluye un ejemplo en el Anexo D) para comprobar su correcto funcionamiento, así como el también correcto envío de la información al CMS.

La calidad del sensor DHT22 no es excesivamente alta, requiriendo tiempos de muestreo elevados, lo que provoca que cada cierto tiempo haya errores en la lectura de los datos. Para solucionar este problema se ha realizado una pequeña modificación en el driver Adafruit_DHT_Driver, de forma que cuando se produzca el error en la lectura, devuelva el último valor registrado.

3.2. Transmisión de datos

Constituye el siguiente paso de diseño, tras la captura de los mismos. A continuación, se analizan los aspectos más relevantes que conciernen a esta funcionalidad. Se comienza con el estado del arte, donde se ofrecen un conjunto de conocimientos necesarios para entender los conceptos relativos a la sección que precede.

Se continúa con el diseño de la transmisión de datos, donde a través de los diagramas correspondientes se describe de manera conceptual la dinámica de comunicación entre drivers y CMS. Por último, está la implementación, donde explicamos cómo se ha desarrollado esta fase siguiendo los mencionados diagramas.

3.2.1. Estado del arte.

EmonCMS hace uso de un identificador conocido como API Key. Existen dos tipos de API Key:

- Read API Key: permite la lectura de datos del CMS, pero no su modificación.
- Write API Key: se permite tanto la lectura como la escritura de los datos del CMS.

La figura 13, muestra un ejemplo de las API Key que tiene nuestro sistema.

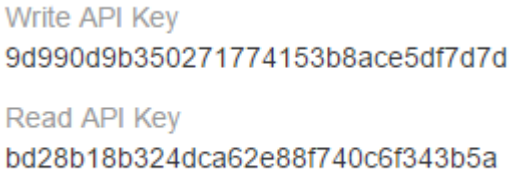


Figura 13: Ejemplo de API Key

Con este identificador, se puede hacer uso de la API de emonCMS, la cual es necesaria para el envío de los datos capturados por los sensores a la aplicación, encargada de su tratamiento y procesamiento.

El formato de la llamada a la API, para transmitir y enviar el valor correspondiente es:

```
http://UBICACION/input/post.json?node=NumeroDeNodo&json={NombreDeVariable:ValorDeVaria
ble }&apikey=WriteAPIKey
```

3.2.2. Diseño

En esta sección, se describe la relación de los drivers con la aplicación, así como la representación que ha sido diseñada para la entidad en la base de datos.

3.2.2.1. Comunicación API

El diagrama de la figura 14, representa la comunicación del driver de Python, enviando los datos del sensor, con la API del gestor de contenido que a su vez guarda esos datos en la base de datos.

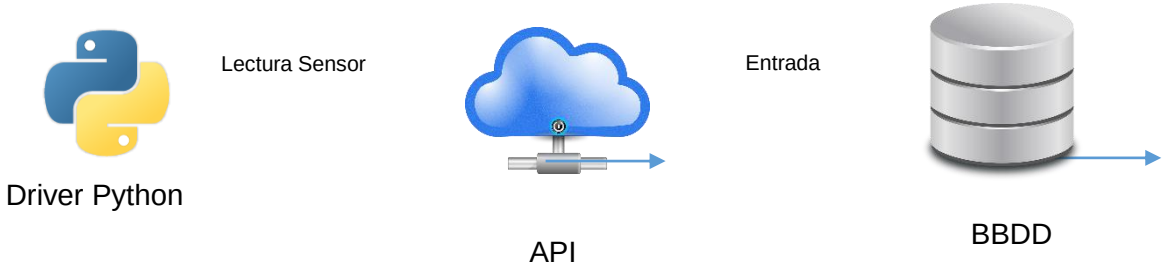


Figura 14: Diagrama de comunicación de la aplicación

3.2.2.2. Entidad Entrada

La figura 15, muestra el diagrama relativo a la base de datos de la entidad Input.

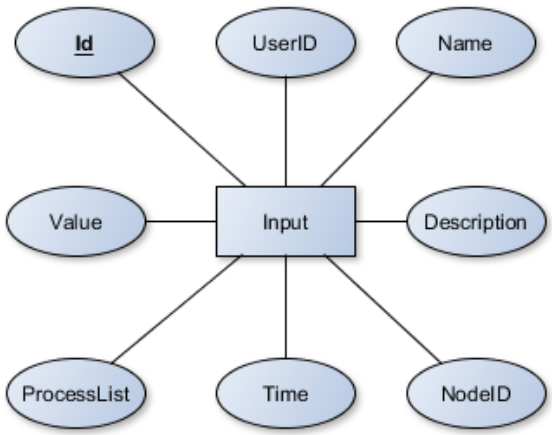


Figura 15: Diagrama de base de datos entidad entrada

3.2.3. Implementación

Para el envío de los datos del sensor, se han modificado sus respectivos drivers escritos en Python. El envío de los datos, se realiza a través del protocolo HTTP, haciendo uso de una librería de Python llamada urllib. Con el fin de solventar el problema de lectura del DHT22, se escribió un algoritmo que toma un valor inicial estándar para temperatura y otro para humedad y lo sustituye por la primera lectura válida que recibe, de forma que, incluso si la lectura falla, en la primera iteración siempre se envía un valor válido al CMS. Se usa el formato JSON para la transmisión de los datos. En la llamada se especifica la dirección de la API de entrada de datos, el número de nodo y el nombre y valor de las variables a enviar.

Una vez ejecutados los drivers, es posible visualizar desde el CMS que la transmisión se ha realizado correctamente. La figura 16, muestra un ejemplo esquemático sobre la visualización de dicho proceso.

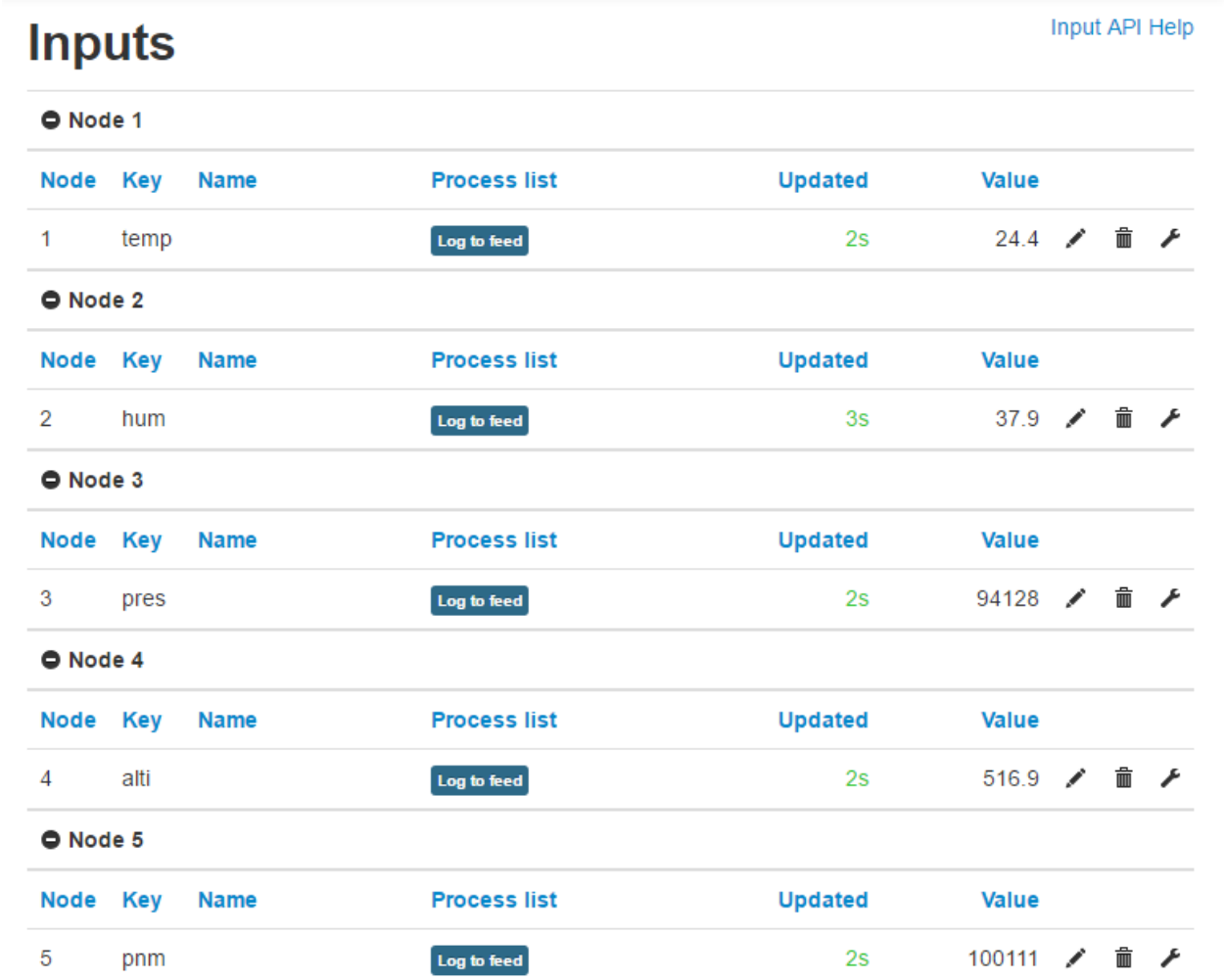


Figura 16: Ejemplo de inputs en el sistema

3.3. Análisis de datos

Constituye la tercera fase del proyecto. En esta sección, se describe todo el proceso necesario para el análisis de los datos recogidos en la fase previa.

3.3.1. Estado del arte.

EmonCMS

El diseño básico de emonCMS consiste en un conjunto de código HTML, CSS y JavaScript en el lado del cliente que sirve como interfaz de usuario, siendo cargada desde el servidor cuando la página es inicialmente requerida.

La parte del cliente envía peticiones a la API a través de AJAX, con un intercambio de datos en formato JSON. La API del servidor, es una API HTTP, que referencia a modelos internos que se encargan del almacenamiento de datos, procesamiento y validación.

En la figura 17, se muestra la estructura de ficheros y directorios que forman la base del CMS, cada uno de sus módulos sigue el patrón MVC. Estos son algunos de los ficheros que contiene, aquellos que se consideran más representativos.

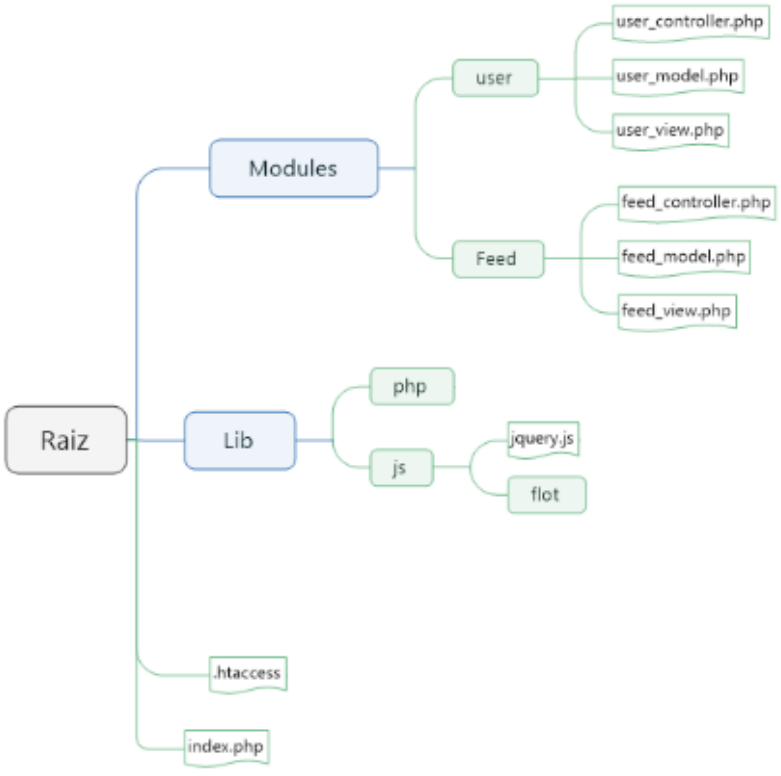


Figura 17: Árbol de directorios principal de emonCMS

La estructura del CMS que conforma el presente proyecto, está diseñada bajo el bien conocido patrón modelo–vista–controlador (MVC) [23], asegurando así la modularización de sus componentes. Cada uno de sus módulos a su vez sigue este patrón de diseño.

El MVC [23], es un patrón de arquitectura software, que separa los datos y la lógica de negocio de una aplicación, de la interfaz de usuario, y del módulo encargado de gestionar los eventos y las comunicaciones. Para ello, MVC propone la construcción de tres componentes distintos, que son, el modelo, la vista y el controlador, es decir, por un lado, define componentes para la representación de la información, y por otro lado para la interacción con el usuario

De manera genérica, los componentes de MVC se definen en los siguientes puntos y en el esquema de la figura 18:

- Modelo: es la representación de la información con la cual el sistema opera, por lo tanto, gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la 'vista' aquella parte de la información que, en cada momento, se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información, llegan al 'modelo' a través del 'controlador'.

- Controlador: responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se realiza alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar comandos a su 'vista' asociada, si se solicita un cambio en la forma en que se presenta el 'modelo' (por ejemplo, desplazamiento o scroll por un documento o por los diferentes registros de una base de datos). Por tanto, se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo' (véase Middleware).

- Vista: presenta el 'modelo' (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario), por tanto, el propio usuario requiere de dicho 'modelo' la información que debe representar como salida.

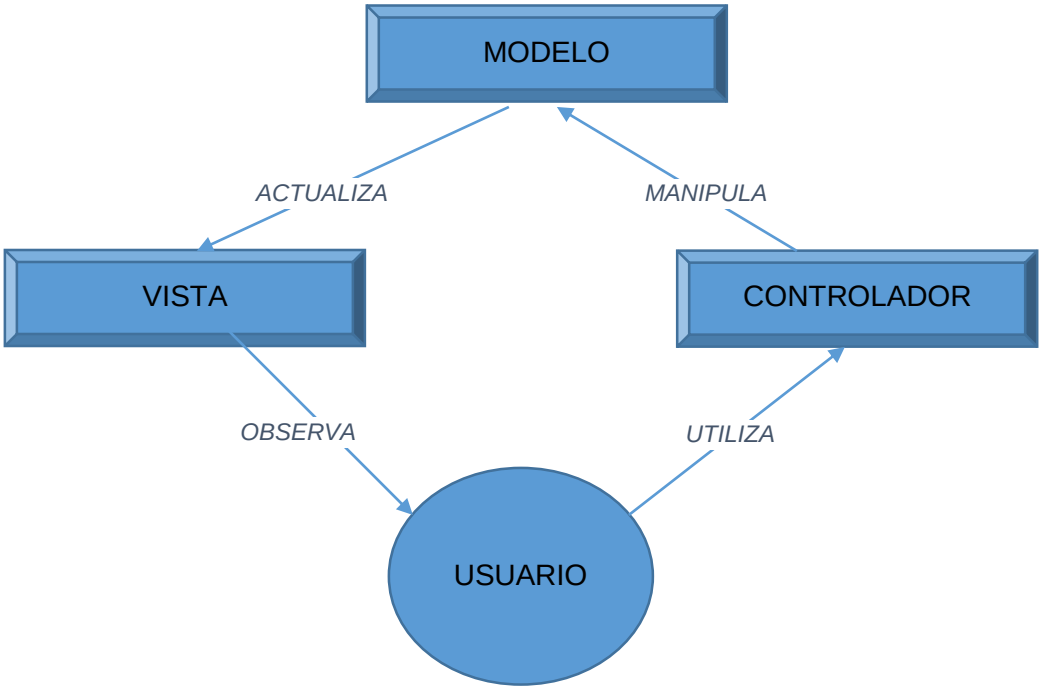


Figura 18: Esquema MVC

EmonCMS está dividido en módulos con una estructura definida, de forma que agregar nuevas funcionalidades sea más fácil.

Un módulo, es simplemente un directorio con todos los archivos que correspondan a una funcionalidad interna. Los módulos principales de emonCMS son: *user, input, feed, vis y dashboard*.

Cada módulo esta subdivido siguiendo el patrón MVC de la siguiente forma:

- Controlador del módulo: consta de una segunda fase para el manejo de las peticiones vía HTTP, después de un pre proceso por parte del index.php. Se dispone de una pequeña estructura para las peticiones que consta de acción, subacción y formato, cuya finalidad consiste en simplificar el manejo de las peticiones entrantes.

- Modelo del módulo: es una clase que representa una entidad de la aplicación. Se encarga de mantener la estructura de las propiedades de la entidad, de cómo se almacena en la base de datos y de las verificaciones propias relativas a la estructura que almacena.

- Vista del módulo: es la encargada de interactuar con el usuario, haciendo uso de tecnologías web tales como HTML, CSS, JAVASCRIPT, etc. Sirve de intermediario entre el usuario y el controlador de la aplicación.

Inteligencia Artificial

El concepto de Inteligencia Artificial (IA) [6] [7] a veces resulta un tanto difuso. Para poder hablar de la IA, primero es necesario determinar qué se entiende por *Inteligencia*, lo cual se puede definir como “la capacidad de adquirir y aplicar conocimiento” y por otro lado, qué es el *comportamiento inteligente*. Por poner un ejemplo sencillo: el hecho de ser capaz de recopilar información, y a partir de ella, deducir un diagnóstico, supone inteligencia.

Llevar el razonamiento que puede tener cualquier persona, a una máquina, es una de las cuestiones más complejas de modelar que existen en el campo de las ciencias de la computación.

Para una persona, aplicando el sentido común, a menudo resulta suficiente como ayuda para prever multitud de hechos y fenómenos corrientes, pero que, como ya se ha indicado, resulta muy complicado su representación en un ordenador ya que los razonamientos, son en muchas ocasiones inexactos, dado que sus conclusiones y reglas en las que se basan solamente son aproximadamente verdaderas.

Según Russell y Norvig [24], se pueden diferenciar diferentes tipos de IA:

- Sistemas que piensan como humanos: tratan de emular el pensamiento humano; por ejemplo, las redes neuronales artificiales. También encajan aquí la automatización de actividades vinculadas con procesos de pensamiento humano, incluyendo actividades como la toma de decisiones, resolución de problemas y aprendizaje.

- Sistemas que actúan como humanos: tratan de actuar imitando el comportamiento humano; por ejemplo, la robótica.

- Sistemas que piensan racionalmente: es decir, con lógica, tratan de imitar o emular el pensamiento lógico racional del ser humano; por ejemplo, los sistemas expertos.

- Sistemas que actúan racionalmente: tratan de emular de forma racional el comportamiento humano; por ejemplo, los agentes inteligentes. Está relacionado con conductas inteligentes en distintos dispositivos y artefactos.

El módulo de IA desarrollado para esta aplicación, está englobado dentro de los sistemas que piensan como humanos, ya que la finalidad que persigue es la ayuda a la toma de decisiones, en nuestro caso, para elegir la vestimenta adecuada según las condiciones atmosféricas o climatológicas en el momento de la interpretación de la información, una vez convenientemente procesada.

El aprendizaje automático, es una rama de la inteligencia artificial que tiene como objetivo desarrollar técnicas que permitan a las computadoras "aprender", es decir crear programas

capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos

La figura 19, muestra un esquema gráfico representando la generalización de las fases asociadas a un proceso de aprendizaje automático. En él, se distinguen dos fases bien diferenciadas, a saber “off-line y “on-line”, ambas con procesos comunes. Así, tras la captura de información por parte de los sensores, a partir de ella se extraen las propiedades pertinentes consideradas relevantes según el objetivo de aprendizaje. Esta información se codifica convenientemente, de forma que sea entendida y asumida por los procesos posteriores (Aprendizaje, Decisión). Estos procesos son comunes a ambas fases. Los datos previamente codificados, son proporcionados al módulo de Aprendizaje durante la fase off-line, el cual aplica el método apropiado para extraer la información relevante, normalmente en forma de parámetros de aprendizaje que se almacenan en la Base de Datos, o más propiamente en la Base de Conocimiento (BC). En el momento de tomar una decisión, con los datos disponibles en el mismo instante de la acción y también con los parámetros almacenados en la BC, se elabora el razonamiento más apropiado para tomar la decisión en cuestión, acción que se realiza durante la fase “on-line”.

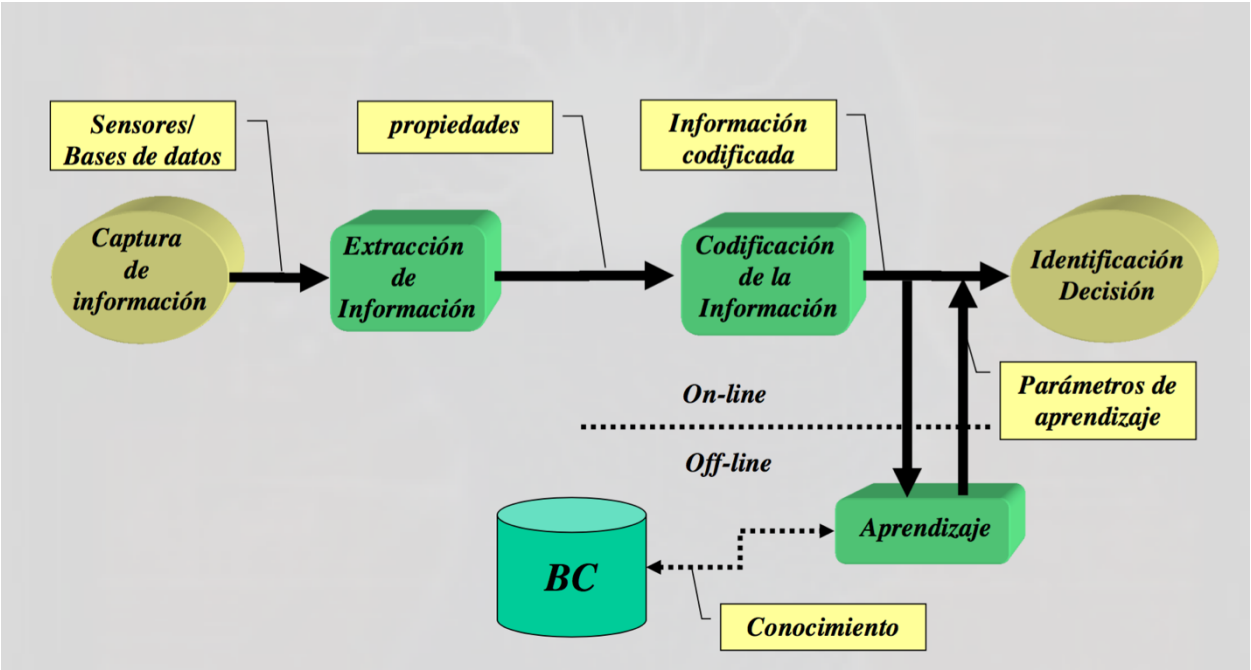


Figura 19: Esquema Aprendizaje Automático [25]

En este proyecto se ha decidido utilizar como método de aprendizaje automático [25], el clasificador paramétrico de Bayes, también llamado Clasificador bayesiano en una de sus versiones más simples [26].

Este algoritmo, es un clasificador probabilístico de naturaleza estadística, fundamentado en el teorema de Bayes [26], como su nombre indica, permite asignar datos a clases, según una estructuración previa de dichas clases.

Por poner un ejemplo, en Pajares [28] [29] se propone una clasificación de las texturas presentes en las imágenes por su color. Se establecen cuatro clases asociadas a otros tantos colores. A cada clase se le asignan píxeles representativos del color que definen. Esos píxeles poseen tres valores, según sus componentes R, G y B. Con todos los píxeles pertenecientes a la misma clase se obtiene un representante de la misma, que no es ni más ni menos que su media aritmética, junto con una medida de la dispersión de los valores de los píxeles en la clase, esto es la matriz de covarianza. Éstos son los parámetros aprendidos durante la fase de entrenamiento o aprendizaje on-line mencionada previamente. Con estos parámetros y en función del cálculo de la denominada matriz de Mahalanobis se determina en función de un criterio de mínima distancia la probabilidad de pertenencia de cada nuevo píxel a una determinada clase durante el proceso de decisión on-line también mencionado previamente.

3.3.2. Diseño

A continuación, se describe el caso de uso “nueva clase” mediante los correspondientes diagramas de actividad, clase y secuencia, que permite generar la estructura de clases requerida por el clasificador de Bayes para su integración en la aplicación

3.3.2.1. Diagrama de actividad

El diagrama de actividad, describe de forma abstracta el procedimiento a seguir en un caso de uso determinado. El usuario introduce el nombre de la clase a agregar, si la clase ya existe se le vuelve a solicitar. Si no existe se procede a leer los feeds, la lectura de feeds es un acceso a la base de datos donde consultamos los valores que nos han dado nuestros sensores. La base de datos se actualiza insertando esta nueva clase en el campo de contenido del algoritmo. Si no hubo ningún error, mostramos mensaje de confirmación, en caso contrario, mostramos mensaje de error. En la figura 20 se detalla el caso de uso “Insertar nueva clase”.

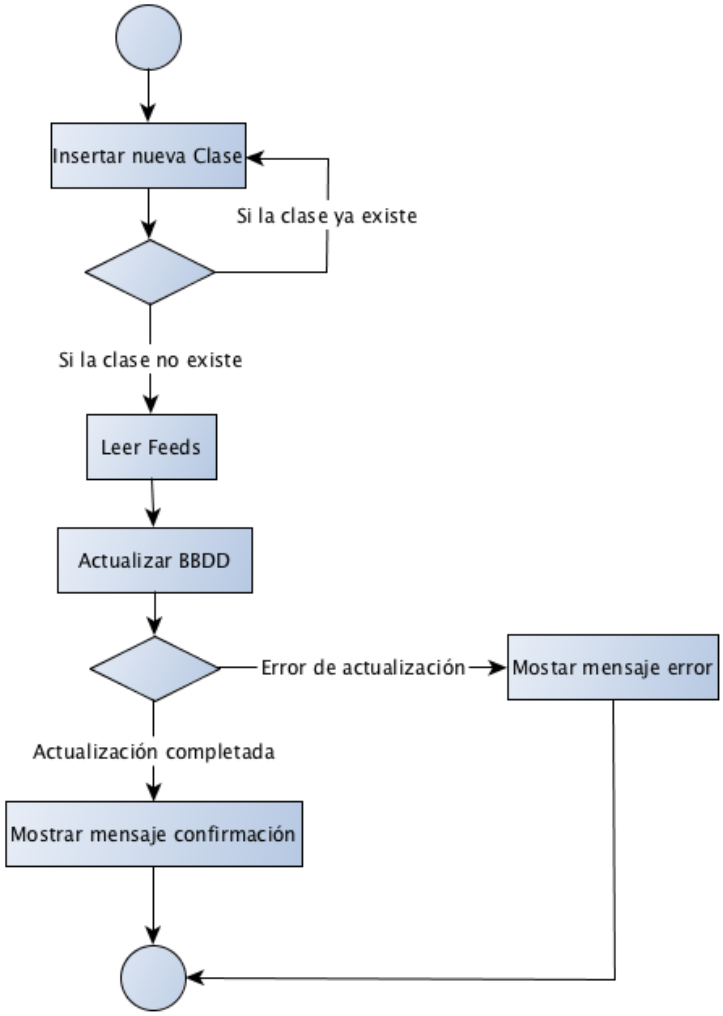


Figura 20: Diagrama de actividad para caso de uso nueva clase

3.3.2.2. Diagrama de clase

El diagrama de clases describe la estructura interna de las clases y su interacción entre ellas. "`IAkairos_view_bayes.php`" se encarga de la visualización y la recepción de acciones del usuario. Hace uso de "`Bayes_algorithm.js`" que se encarga de hacer los cálculos necesarios propios del algoritmo y mantenerse sincronizada con su entidad en la base de datos. Para ello hace uso de "`IAEntity.js`", que se encarga de mandar las peticiones del algoritmo al controlador. El controlador utiliza el modelo, en nuestro caso "`IAkairos_model.php`" para interactuar con la base de datos.
En la figura 21, se muestra la relación de uso de las clases propia del MVC.

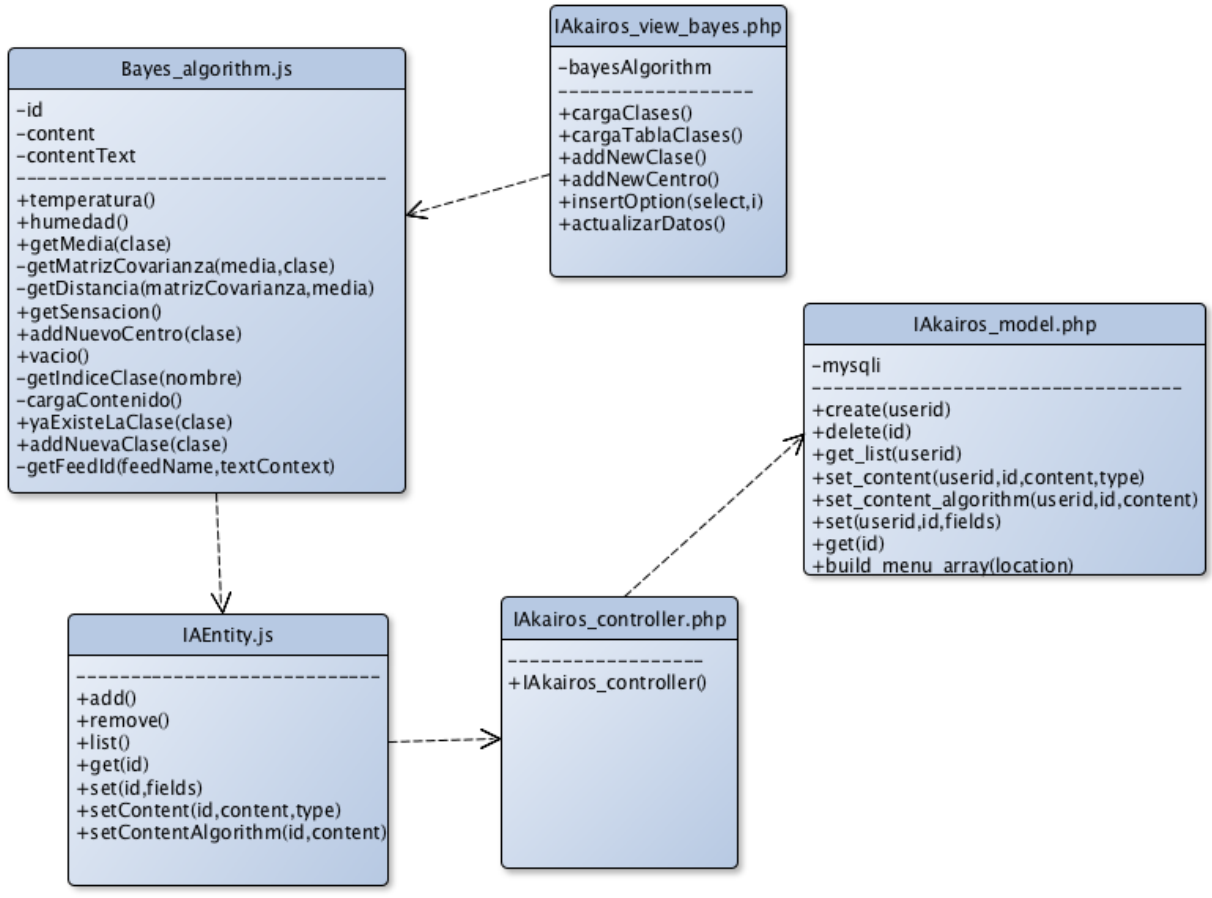


Figura 21: Diagrama de clase caso de uso nueva clase

3.3.2.3. Diagrama de secuencia

Un diagrama de secuencia muestra la secuenciación, en este caso el flujo de acciones relativas al envío de mensajes entre las distintas clases relacionadas con un caso de uso.

Cuando el usuario accede a la vista, esta genera una instancia de `bayes_algorithm.js` que, a su vez, al inicializarse, le pide a `IAEntity.js` que le devuelva todos los datos relativos a su entidad. Para ello `IAEntity.js` hace un `get` con el identificador del IAEntity que hemos visualizado haciendo uso del controlador, y este a su vez llamando al modelo para que le proporcione los datos.

Una vez instanciado el algoritmo, la vista le pide los datos actuales de temperatura y humedad para mostrarlos, y le pregunta si el algoritmo tiene contenido. En caso de tener contenido, se muestra una predicción en base a ese contenido, en caso contrario, se muestra un mensaje pidiéndole al usuario una clase nueva.

Para este caso de uso en concreto se hace una petición para generar una nueva clase.
La vista invoca un método propio llamado "`addNewClase`". Si no dispone de ningún dato previo, significa que no existe ninguna clase previa, por lo tanto, procedemos del mensaje que mencionamos anteriormente.

Para este caso, una vez añadida la nueva clase, deberemos ocultar el mensaje anterior. Si, por el contrario, ya disponíamos de una clase previa, simplemente ocultaremos el desplegable que cuelga de la opción "No" y "añadir nueva clase".

Desde ese método propio de la lista se invoca a un método del algoritmo `bayes_algorithm.js` que se encarga de añadir una nueva clase con las lecturas de los sensores actuales. Se vuelca esta nueva información haciendo uso del `IAEntity.js`, éste a su vez llama al controlador y por último el controlador envía el mensaje al modelo.

En la figura 22, se describe el comportamiento de la aplicación y su interacción con otras clases relevantes para el proceso.

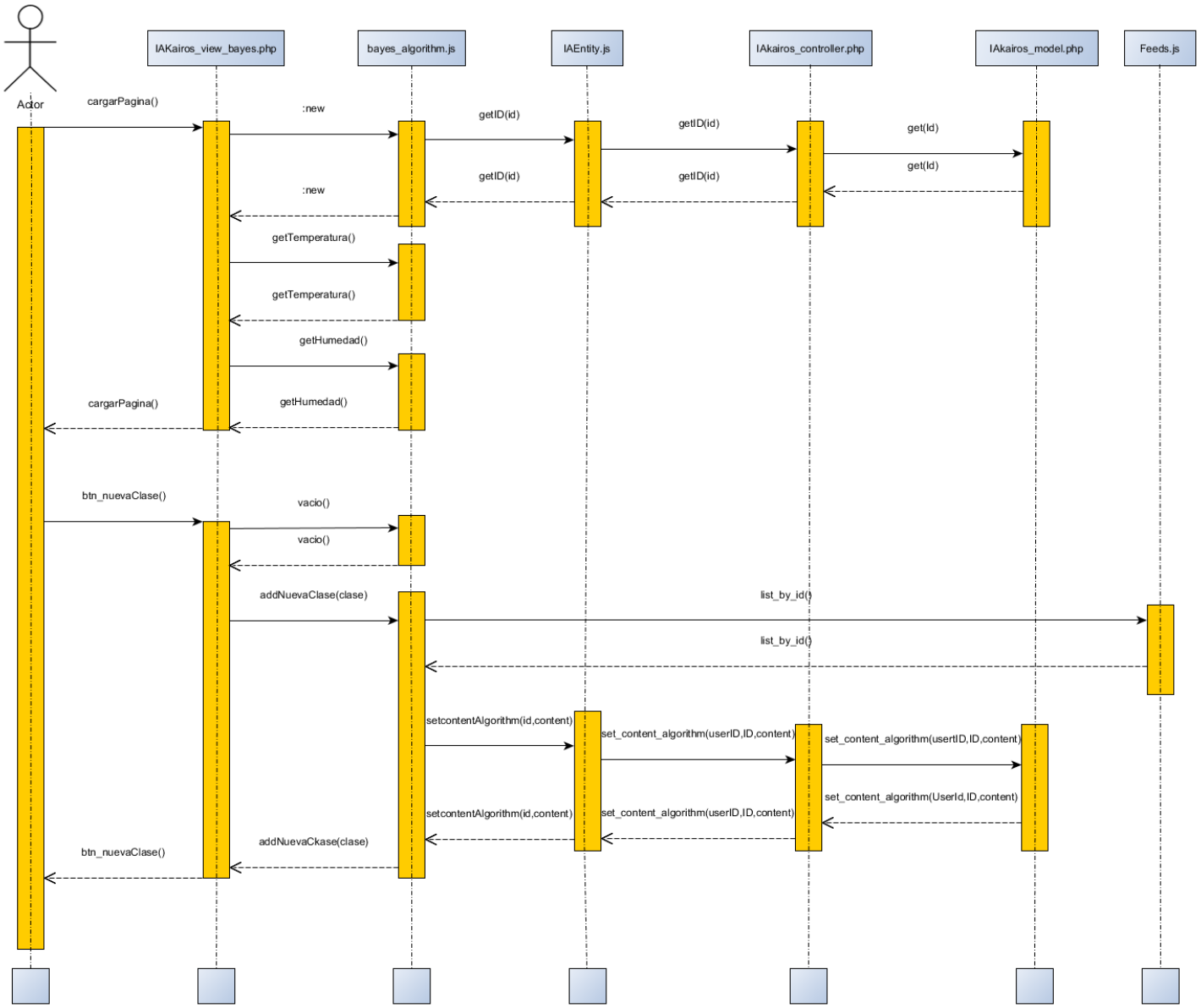


Figura 22: Diagrama de secuencia caso de uso

3.3.3. Implementación

La figura 23, muestra el árbol de directorios que describe los ficheros importantes de la aplicación. La raíz contiene los ficheros que forman la base indispensable del gestor de contenido. Además, contiene una carpeta llamada modules que contiene todos los módulos que incrementan la funcionalidad del sistema. Cada módulo sigue el patrón MVC, por tanto, encontramos archivos que implementan la vista, el modelo y controlador. Dentro de nuestro modulo, se encuentran el controlador, el modelo, una vista general, una carpeta con los algoritmos y otra con las vistas. En las vistas tenemos la configuración, las listas, y las respectivas interfaces de cada algoritmo. En la carpeta de algoritmos, tenemos las implementaciones de todos los algoritmos que se han desarrollado. En nuestro caso se tiene Bayes y Lloyd, sin embargo, el de Lloyd es sólo una copia del algoritmo de Bayes que se utiliza exclusivamente con fines de pruebas.

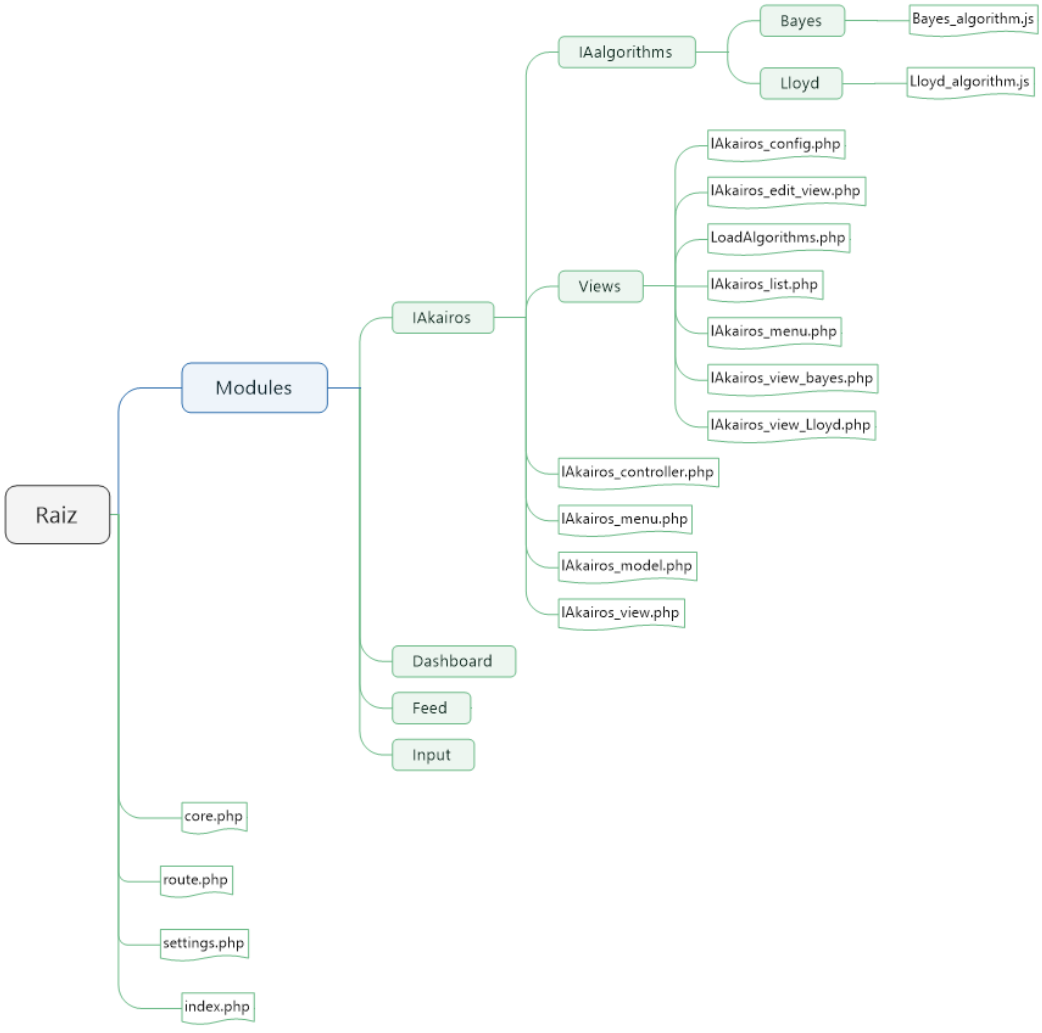


Figura 23: Árbol de directorios de la aplicación

3.3.3.1. Modelo

En el modelo se ha implementado una clase llamada IAEntity, que se encarga de mantener la estructura de la entidad de IA ofreciendo la funcionalidad de un “transfer” y la integración con la capa de almacenamiento, en nuestro caso la BBDD. Un transfer, es un patrón de diseño que encapsula todos los campos característicos de la entidad que representa, a fin de facilitar su uso en la aplicación.

Con la clase IAEntity, se pretende simular la funcionalidad de la clase implementada en emonCMS conocida como Dashboard, lo que nos permite generar instancias de algoritmos completamente distintos. La IAEntity representa un algoritmo de inteligencia artificial cualquiera, y puede configurarse con uno de los algoritmos que haya implementados en el sistema.

Como se ha indicado previamente, para este proyecto se ha desarrollado un clasificador paramétrico de Bayes. Sin embargo, la flexibilidad de diseño del sistema, tal y como se ha concebido, permite, sin apenas cambios, generar un archivo que describa el comportamiento y las opciones de configuración necesarias para un nuevo algoritmo (existente en el sistema) de forma que esta operación se realice automáticamente para agregarlo a la estructura del mismo. En la figura 24, se muestra un ejemplo de la implementación de esta entidad, junto con los iconos de selección, edición y eliminación.

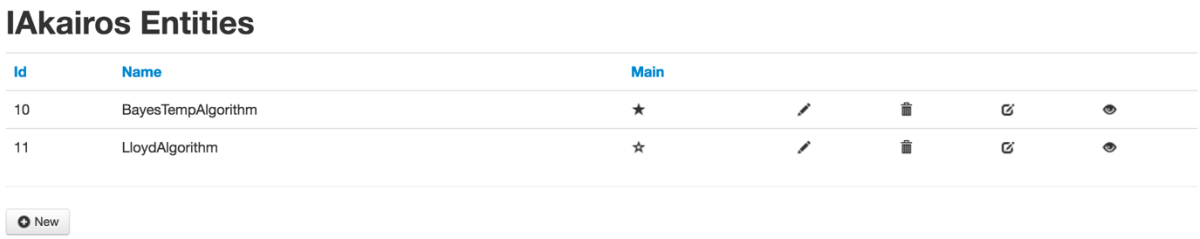


Figura 24: Ejemplo de entidades

Se ha desarrollado la página de configuración de IAEntity siguiendo la dinámica de configuración de los widgets proporcionados por el CMS. La página de configuración hace uso de una clase en PHP que carga todos los ficheros que sigan un patrón determinado, en nuestro caso `nombreAlgoritmo_algorithm`, devolviendo una lista con todos los algoritmos presentes en el sistema. A continuación, se muestra una lista desplegable, permitiendo seleccionar uno de los algoritmos de aprendizaje/clasificación existentes. Dependiendo del algoritmo seleccionado se realiza una llamada a una función presente en su implementación en JS que devuelve todas las opciones requeridas por ese algoritmo en concreto.

3.3.3.2. Vista

La vista es específica para cada algoritmo de IA implementado. Debido a que la funcionalidad que tiene cada algoritmo puede diferir en gran medida, no resulta apropiado crear una vista generalizada o común. En nuestro caso, se realizó un diseño web con una estética limpia, atractiva y totalmente funcional, con contenido dinámico. En adelante se hace referencia a la vista del algoritmo implementado, esto es, un clasificador bayesiano, cuya descripción general sería válida para cualquier otro existente en el sistema

La idea principal de la vista, es la de diseñar una interfaz, que proporcione información útil, más específicamente la temperatura y la humedad actual como datos proporcionados por los sensores y por supuesto la sensación térmica que tendrá el usuario según los factores ambientales actuales proporcionados por el módulo de IA una vez tomada la decisión correspondiente. Para ello, al cargar la página, se actualizan los valores de temperatura y humedad, que se muestran, comprobando si existen datos de entrenamiento previos, almacenados en la BC, de forma que, en caso de existir, se realiza la predicción apropiada. Si no existen datos almacenados, lo que significa que no se ha realizado aprendizaje previo, se procede a mostrar un menú distinto al anterior donde se invita al usuario a establecer su primera clase (categoría), tras lo cual se introduce esa nueva clase con los datos asociados a las lecturas de los sensores en ese momento.

Una vez existe suficiente información almacenada en la BC para realizar una predicción, se muestra ésta y se ofrecen las opciones “Sí” y “No”. Si se selecciona la primera opción se activa el algoritmo correspondiente y se registran las lecturas actuales del sensor con la clase mostrada como predicción para continuar aprendiendo en base a más datos y más clases. En caso de seleccionar la segunda opción se desplegará un menú donde se puede seleccionar una de las clases ya existentes o bien añadir una nueva en caso de que la situación actual no coincida con ninguna de las opciones previamente aprendidas. En este sentido, se puede afirmar, que se trata de un mecanismo de aprendizaje incremental.

3.3.3.3. Controlador

El controlador es relativamente simple en la presente aplicación. Su función es cargar la página a la que quiere acceder el usuario. Recibe una estructura generada por el `index.php`, y dependiendo de los valores que ésta tenga, carga una página u otra, a la vez que solicita los permisos específicos correspondientes, consistentes básicamente en escritura o lectura para edición o para lectura. Si la sesión cuenta con permisos de lectura, páginas como la de edición de algoritmos no podrán ser visualizadas. Otra función importante del controlador es la de comunicar las peticiones realizadas por la vista al modelo, de forma que, si se quiere modificar una entidad, esta petición se solicita al controlador de forma que éste a su vez llama al modelo.
La figura 25, muestra el comportamiento del controlador cuando se realiza una petición. Desde la captación y división de la url por parte de `login.php`, hasta su redirección en función de los parámetros recibidos en el controlador.

El usuario introduce la url en su navegador, esta petición es procesada por `Index.php` que subdivide la url en campos más pequeños. Estos campos son `Module` , que indica el módulo al que se quiere acceder, `Action`, que indica la acción que se quiere llevar a cabo dentro de ese módulo, y `Data`, que proporciona las variables adicionales que se han introducido en la url. `Index.php` reenvía estos campos al controlador del módulo especificado, que se encarga de hacer lo que considere necesario con los datos recibidos.

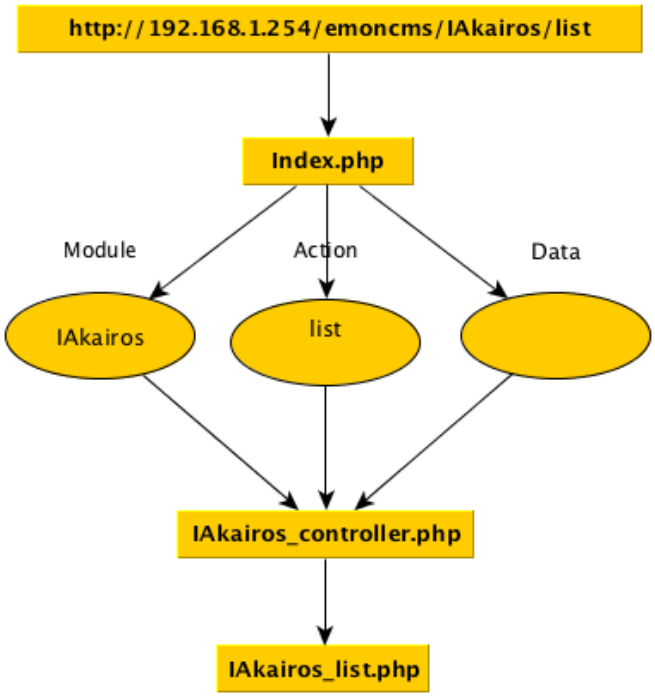


Figura 25: Esquema de funcionamiento del controlador.

4. Resultados

A modo de introducción, en la figura 26, se muestra un esquema conceptual de la aplicación, aquí se pueden observar las diferentes interacciones sobre los distintos componentes de la aplicación, con la Raspberry Pi en el centro del diagrama, es el eje principal de la misma, a ella están conectados los sensores y ejecutándose los drivers que capturan su información, a su vez, está instalado el CMS con todas sus dependencias, base de datos, servidor web, etc. Para que, en remoto, conectada a la nube, los usuarios puedan hacer uso de la aplicación, utilizando los correspondientes paneles de monitorización (dashboard) y el módulo de IA (IAkairos).

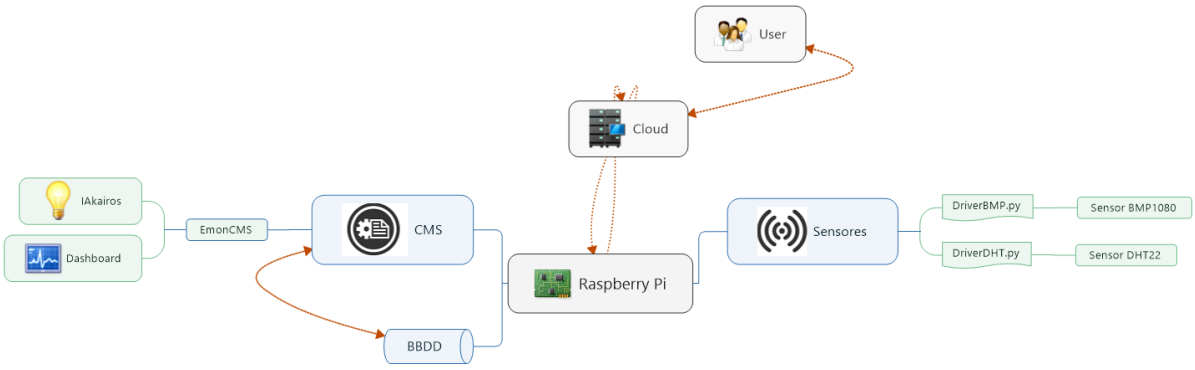


Figura 26: Esquema general de la aplicación

En este apartado, se detallan las partes más relevantes que han sido desarrolladas sin entrar en detalles en lo relativo al funcionamiento desde el punto de vista de un usuario final, ya que este aspecto se detalla en los Apéndices B y C.

Por un lado, se describe e identifica la visualización y monitorización de los datos recogidos mediante los sensores, y por otro lado, el módulo de IA que se ha desarrollado. Ambos pueden verse claramente diferenciados desde el panel de administración del CMS, donde se separan los dos módulos.

Desde el Setup, que es el panel general para administrar la aplicación, se definen los componentes más importantes de la aplicación, aquí tenemos los siguientes módulos;

- Inputs:

  Desde este módulo, podemos visualizar las entradas que nos llegan vía HTTP desde los drivers de los sensores

- Feeds:

  El feed, es lo que maneja y procesa en CMS, para interactuar con ello, tenemos que seleccionar los inputs deseados y agregarlos como feeds,

- Dashboards:

  El módulo de dashboards, nos permite realizar las interfaces gráficas mediante un conjunto de widgets. Estos elementos de visualización se conocen como widgets y son una representación de uno o varios datos que se reciben como feeds.

- IAkairos:

  El módulo IAkairos, es el módulo de Inteligencia Artificial que se ha implementado, en él podremos entrenar el algoritmo de inteligencia artificial que se haya implementado y visualizar sus parámetros, así como su toma de decisiones.

- Módulos de administración y documentación.

La figura 27, muestra el acceso desde el Setup (inicio) a estos módulos, describiéndose claramente la separación entre dichos módulos.

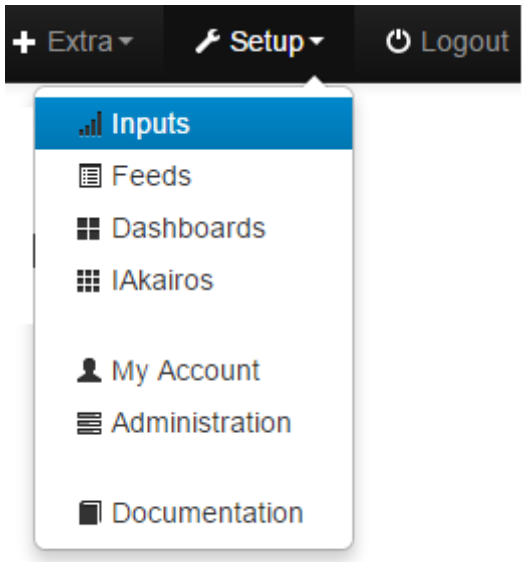


Figura 27: Setup de la aplicación

Con base en el esquema de la figura 27, a continuación, se describen estos cuatro componentes principales.

En el módulo de inputs, se pueden ver las entradas de datos procedentes de los sensores, quedando disponibles para agregarlos a la aplicación como feeds y poder hacer uso de ellos.

En la figura 28, se muestra un simple ejemplo de 2 inputs recibidos en el sistema, que son temp y hum. (temperatura y humedad). Se puede ver en dicha figura cómo vienen por dos nodos diferentes, también su valor y el tiempo transcurrido desde su última actualización.

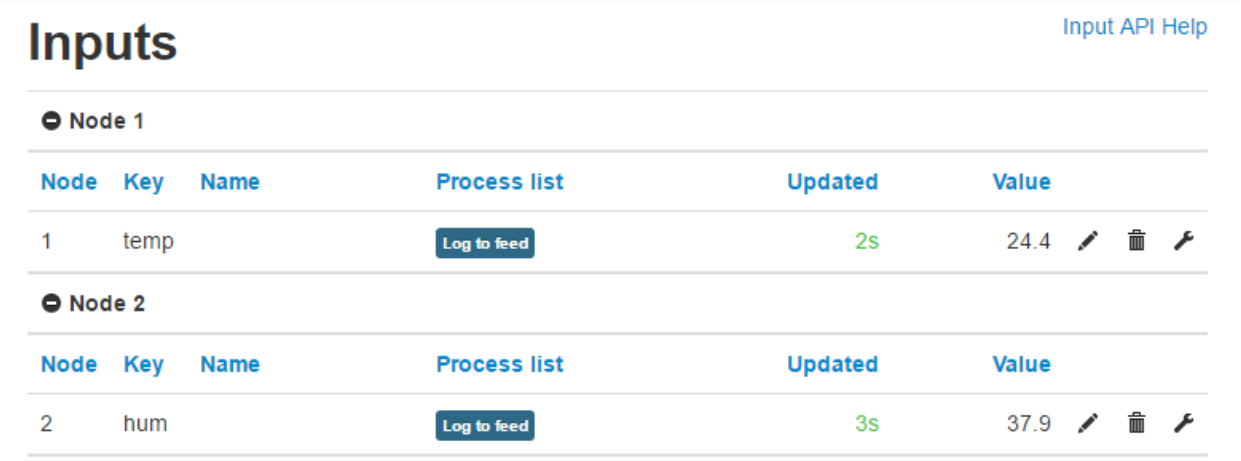


Figura 28: Ejemplo de Inputs en la aplicación

Una vez tenemos los datos disponibles, hay que agregar aquellos que sean de interés como feeds, para que la aplicación pueda utilizarlos y procesarlos.

En la figura 29, se muestra un esquema de los campos asociados con dichos datos, visualizándose el nodo de procedencia y la identificación del dato (temperatura, humedad), el tipo de dato, en este caso Realtime, así como su valor y el tiempo que lleva sin actualizarse. Por otro lado, en su parte derecha, tenemos unos botones de control, para poder visualizar, eliminar o modificar el feed en cuestión.

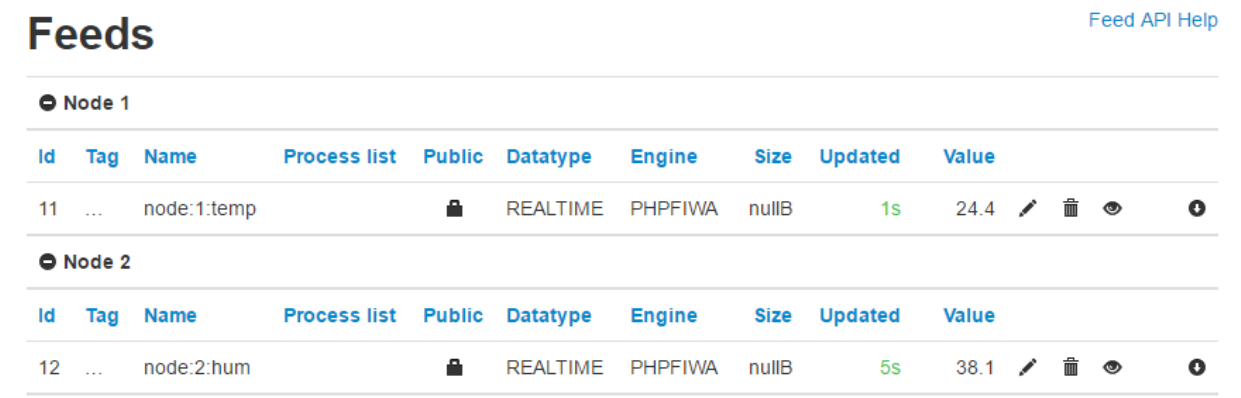


Figura 29: Ejemplo de Feeds en la aplicación

Una vez definidos los feeds a manejar en la aplicación, se puede utilizar esa información como entrada para el módulo de dashboard (visualización) o para el módulo de IA IAkairos.

En primer lugar, se muestra un ejemplo de dashboard, o panel de visualización y monitorización de los datos recogidos, cuyo manejo se puede realizar desde el Setup, accediendo a la pestaña Dashboard, pudiéndose añadir tantos dashboard como se deseen. En la figura 30, se muestra un ejemplo de varios dashboard que se han creado en el sistema.

Como muestra dicha figura, desde este panel, se puede hacer público o dejar en privado un dashboard, así como crear un Alias para un determinado dashboard, que es una referencia más cómoda para acceder a él mediante una url. Por otra parte, tenemos los botones de control, donde se puede, duplicar, editar el nombre o alias, eliminar, edición de los widgets y sus configuraciones, o visualizar dashboards.

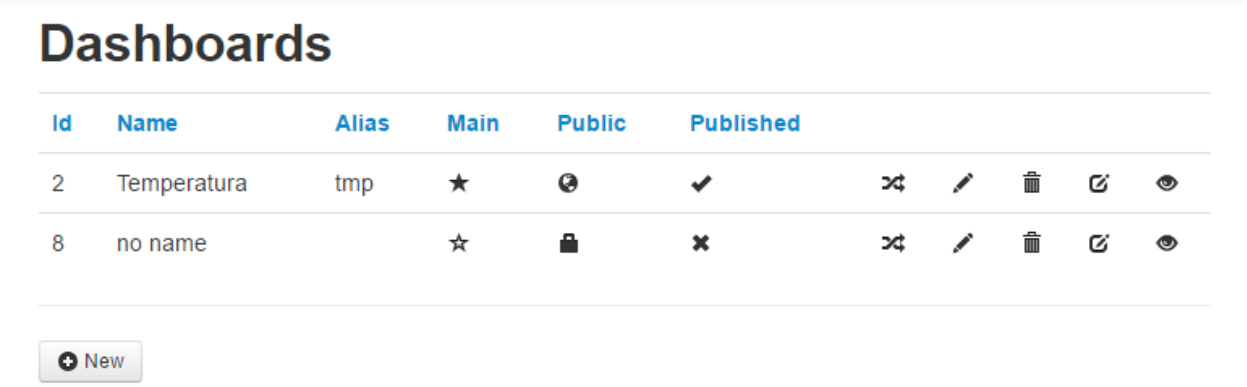


Figura 30: Ejemplo de Dashboards en la aplicación

Para ilustrar de forma representativa lo que se puede realizar en el dashboard, en la figura 31 se muestra un ejemplo completo de un dashboard. Debe tenerse en cuenta que en este ejemplo de dashboard, hay más feeds que los mostrados en ejemplos anteriores, como por ejemplo altitud o presión atmosférica.

Figura 31: Ejemplo de visualización de un dashboard en la aplicación

En esta visualización se tiene, por un lado, en la parte izquierda, los datos de temperatura y humedad actuales, en la parte central, aparecen cuatro gráficas que representan el histórico de los valores de los parámetros asociados (Temperatura, Humedad). Como puede apreciarse, según la resolución seleccionada para cada gráfica, es posible visualizar rangos que varían dese un minuto a una hora, o rangos entre un día y un año. Por último, en la parte derecha, aparecen sendos gráficos que monitorizan la presión atmosférica y presión a nivel del mar en mb y Pa respectivamente.

En la parte de arriba, se ha añadido también los datos de altitud respecto del nivel de mar, así como la temperatura del punto de rocío.

Una vez visto el módulo de dashboards relativo a la visualización de datos, se detalla el módulo de Inteligencia Artificial.

Desde el setup, accediendo a IAkairos queda disponible el panel de administración. Al igual que ocurre con los dashboards, se pueden crear tantos como métodos de aprendizaje tenga implementados la aplicación. La figura 32, muestra la creación de una nueva entidad de esta naturaleza, en este caso Bayes.

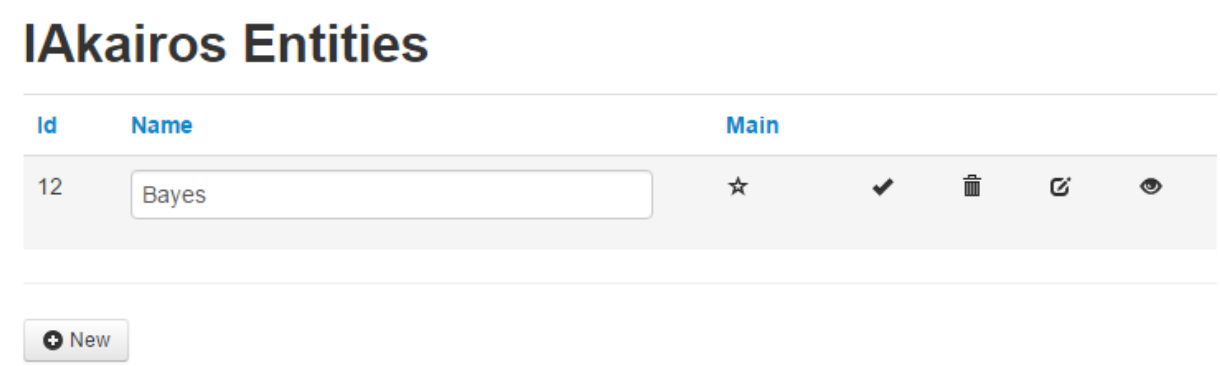


Figura 32: Ejemplo de creación de una entidad en la aplicación

Una vez creada la entidad, es necesario configurar dicho algoritmo con los feeds que se deseen utilizar, exactamente igual que ocurre con los widgets de los dashboard, para ello, se accede a la parte de edición como muestra la figura 33 y se seleccionan los feeds deseados de entre los disponibles.

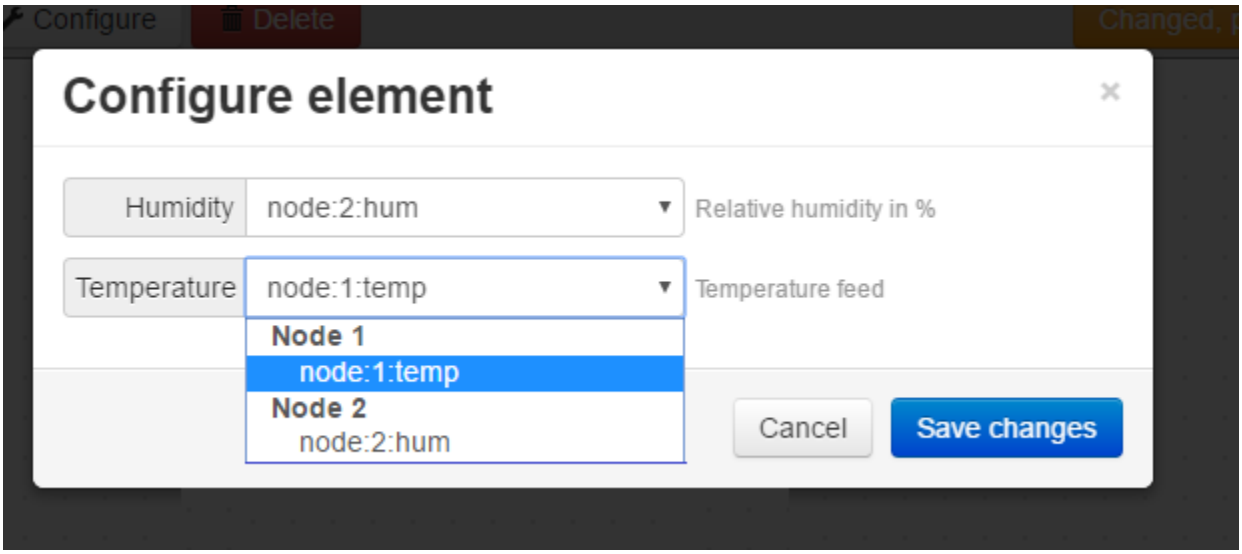


Figura 33: Ejemplo de configuración de feeds en el algoritmo de Bayes

Una vez configurados los feeds requeridos, se puede acceder directamente a su interfaz gráfica, que, en una primera instancia, al no existir datos procesados por el algoritmo de aprendizaje, no existen clases asociadas. En la figura 34, se muestra exactamente dicho proceso, donde, tal y como se puede ver en la figura autoexplicativa, se permite la creación de

la que sería la primera clase asociada al algoritmo, eligiendo los parámetros actuales como el primer representante de la clase, al que se le añadirán nuevos patrones de muestra a medida que el método vaya procesando información y aprendiendo.

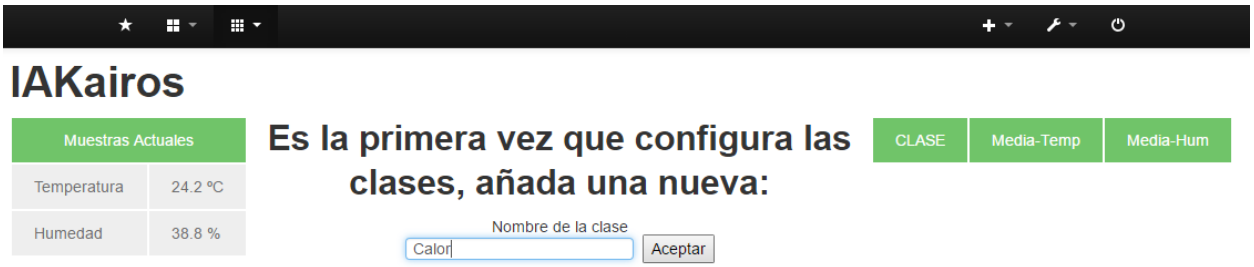


Figura 34: Ejemplo de interfaz gráfica sin clases definidas

Una vez creada la primera clase en el sistema, ésta se añadirá en la tabla de la derecha, donde se muestran todas las clases existentes hasta el momento en la aplicación manejadas por el algoritmo, con las medias de sus centros. La figura 35, muestra exactamente cómo quedaría la vista cuando se crea la clase Calor con los datos disponibles.

Figura 35: Ejemplo de interfaz gráfica con clases definidas

Llegados a este punto, es el momento de proceder a proporcionar datos al sistema para el aprendizaje. Estos datos se introducen en función de la sensación climatológica de cada persona. En efecto, si una persona para unos valores determinados en los parámetros meteorológicos, tiene la sensación de calor, dichos datos se añadirán a la clase correspondiente. Un usuario diferente, podría percibir una sensación distinta con esos mismos datos, por ejemplo, templado, con lo cual, otro usuario, desde su interfaz, podría definir sus propias clases. Esto significa que el sistema funciona bajo una perspectiva personalizada en función de las sensaciones de cada usuario, hecho diferencial que se ha contemplado en la aplicación para no mezclar datos de distintos usuarios que han percibido distintas sensaciones con idénticos datos.

La aplicación permite verificar que la sensación percibida es la correcta, bajo los criterios personales de los usuarios. Se ofrecen las opciones pertinentes para seleccionar las opciones sí o no, que permite verificar que la sensación es la correcta o por lo contrario, cambiar a otra clase o crear una nueva.

A medida que se van incorporando nuevos datos, el sistema está diseñado para iniciar el correspondiente proceso de aprendizaje de forma automática, permitiendo así la actualización de los parámetros derivados del aprendizaje a medida que los datos se van añadiendo a la aplicación. En este sentido, el sistema realiza un aprendizaje incremental con la incorporación de nuevos datos.

5. Conclusiones y trabajo futuro

Llegados a este punto, se está en condiciones de afirmar que se ha cumplido con el objetivo principal del proyecto. Se ha desarrollado un sistema capaz de visualizar factores ambientales captados remotamente. Básicamente se ha desarrollado una aplicación bajo el paradigma del IoT. Se ha creado una estructura que permite la implementación de distintos módulos, incluyendo los correspondientes a IA de una manera rápida y sencilla gracias al uso de patrones de diseño. El código generado a lo largo de la vida del proyecto es altamente reutilizable y flexible, lo que facilita la creación de distintas vías y oportunidades de desarrollo para continuar con el trabajo realizado en futuros desarrollos. Tras haber estudiado y trabajado con la estructura de emonCMS, se puede afirmar que ha sido un completo acierto el hecho de haber elegido este gestor de contenido. Su estructuración en MVC y su sistema de widgets ha permitido, usando un esquema similar, desarrollar el sistema global y más específicamente la inclusión de algoritmos de IA más ágilmente.

Hubiese sido deseable modificar la página de configuración del algoritmo con mayor detalle para una mejor adecuación a las propiedades que realmente debería tener, por ejemplo, sustituyendo el editor gráfico que está pensado para incluir varios componentes gráficos, en simples opciones que permitan configurar cada algoritmo en concreto. Se ha utilizado la metodología de programación orientada a objetos con Javascript, pese a no ser un lenguaje diseñado específicamente para ello. A pesar de lo cual, la integración del controlador con la vista ha resultado simple, fácilmente legible y modular.

La resolución de la predicción de la sensación térmica mediante el clasificador bayesiano ha dado buenos resultados, devolviendo predicciones bastante acertadas a medida que se entrenaba el sistema con los datos proporcionados al algoritmo. Otra posibilidad de cambio o ampliación futura es la modificación de la BD. Los desarrolladores de emonCMS utilizan una base de datos con el engine MyIsam. Este motor minimiza el tiempo de procesamiento en búsquedas, pero al mismo tiempo carece de un sistema de relaciones entre las entidades. Lo que significa que no se pueden definir claves foráneas, las cuales tienen un papel importante en el mantenimiento de la consistencia de datos. Se decidió mantener el mismo motor a fin de que la integración del código fuese lo más simple posible con el CMS. Se descartó el diagrama de entidad relación original y se usó una estructura un tanto más rudimentaria con el fin de mantener la cohesión entre el código original y el nuevo.

Una nueva línea de desarrollo, debería ser la integración del sistema con otros servicios, satisfaciendo así la cuarta fase del IoT. Una idea que se planteó, y queda pendiente para el futuro, fue la de integrar los datos recogidos por los sensores con un servicio de software libre como OpenStreetMaps [30].

OpenStreetMaps, es un proyecto de software libre que consiste en la elaboración detallada de un mapamundi. La forma en la que estos mapas se muestran depende de la modificación de la última persona que los edite, sólo se necesita una cuenta, pudiéndose modificar, añadir o eliminar cualquier indicación referente a información del mapa de la zona de interés. De esta

forma, es posible avanzar un paso más para conectar la información recogida y analizada por el CMS y sus módulos y permitir a la vez que ese servicio las utilice. Una posible finalidad, podría ser elaborar un mapa de temperaturas, otro de presiones, incluso un mapa con las sensaciones térmicas de las personas, pudiendo resultar curioso observar la variación entre ellas. Incluso crear un sistema que recoja todas las predicciones de cada uno de los sistemas y genere nuevas predicciones más globales.

En este proyecto se han combinado diferentes tecnologías, algunas de ellas ciertamente novedosas. El adquirir los conocimientos necesarios sobre estas tecnologías para desarrollar la idea de proyecto ha sido menos tedioso de lo que pudiera esperarse en un principio. La base de conocimiento proporcionada a lo largo de estos años académicos ha sido de notable utilidad a la hora de abordar problemas desconocidos surgidos en el proyecto. Los paradigmas de la orientación a objetos, la implementación de patrones, mantener los factores de calidad del software han sido piezas claves para poder desarrollar este proyecto.

El IoT parece ser un concepto que va a revolucionar la forma en la que vemos la tecnología en un corto periodo de tiempo, por lo que ha resultado muy interesante trabajar bajo un paradigma de tal naturaleza. Si bien es cierto que las fases de captura y transmisión de datos no encajan con las competencias de un ingeniero de software, por lo menos a niveles de investigación, las fases de análisis de datos y la conexión de servicios son áreas extensas que dan pie a una infinidad de usos para esos datos.

Conclusions and future work

At this point, we may say that the main objective of the project has been accomplished. It has been developed a system capable of displaying remotely captured environmental data. Basically an application under the paradigm of IoT has been developped. It has been created a structure that allows the implementation of different modules, including those for AI, in a quickly and easily way through the use of design patterns. The code generated during the project life is highly reusable and flexible, allowing the creation of different paths and development opportunities to continue the work in future developments. Having studied and worked with the structure of emonCMS it can be said that an appropriate success has been achieved by chosing this content manager.

Its MVC structure and its widget system has allowed, using a similar scheme, to develop the global system and more specifically the inclusion of AI algorithms more nimbly.

It would have been desirable to modify the configuration page of the algorithm in more detail for a better match to the properties that really should have, for example, replacing the graphics editor that is designed to include several graphical components with simple options to configure each algorithm specifically. We used the methodology of object oriented programming with Javascript, although it is not a language designed specifically for it. Despite which, the integration of the controller with the view has proven its simplicity, easily readable and modularity.

The perceptual sensation prediction using the Bayesian classifier has been successful, returning reasonable accurate predictions when the system is trained with the data provided by the user. Another possibility for future expansion or change is the modification of the DB. EmonCMS's developers use a database with MyISAM engine. This engine minimizes the processing time in searches, but it lacks a system of relationships between entities. This means that one cannot define foreign keys, which take an important role in maintaining data consistency. It was decided to keep the same engine in order to preserve the code integration as simple as possible with the CMS. The original entity relationship diagram was discarded and a somewhat rudimentary structure has been used in order to retain the cohesion between the original code and the new used.

A new line of development should be the integration of the system with other services, thus satisfying the fourth phase of IoT. An idea that was raised, and remains for the future was to integrate the data collected by the sensors with a free software service called OpenStreetMaps [30].

OpenStreetMap is a free software project that involves the detailed elaboration of a world map. The way in which these maps are displayed depends on the last update of the person who edits them, you only need one account to modify, add or remove any indication regarding the information map area of interest. In this way it is possible to go one step further to connect the

information collected and analyzed by the CMS and its modules and enable this service to use the data. One possible purpose would be to map temperatures, pressures, describing a map with the thermal sensations of persons. It may be curious to see the variation between them. Even create a system that collects all the predictions of each of the systems and generate new more global predictions.

In this project we have combined several technologies, some of them certainly new. Acquiring the necessary knowledge of these technologies to develop the project idea has been less tedious than might be expected at first. The knowledge base provided along these academic years has been remarkably useful in addressing unknown problems in the project. The paradigms of object orientation programming, the implementation of patterns, the maintaining of software quality factors have been key for developing this project parts.

The IoT seems to be a concept that will revolutionize the way we see technology in a short time period, so it has been very interesting to work under a paradigm of such a nature. While it is true that the phases of capturing and transmitting data do not match the skills of a software engineer, at least to levels of research, phases of data analysis and connection services are extensive areas that give rise to a large number of possibilities.

Apéndices

Apéndice A: Instalación de los requisitos

Vamos a enumerar todas las dependencias que necesitamos instalar en nuestra Raspberry pi para que funcione nuestro CMS.

```
sudo apt-get install apache2
sudo apt-get install mysql-server
sudo apt-get install mysql-client
sudo apt-get install php5
sudo apt-get install libapache2-mod-php5
sudo apt-get install php5-mysql
sudo apt-get install php5-curl
sudo apt-get install php-pear
sudo apt-get install php5-dev
sudo apt-get install php5-mcrypt
sudo apt-get install php5-common
sudo apt-get install php5-redis
sudo apt-get install git-core
sudo apt-get install redis-server
sudo apt-get install build-essential
sudo apt-get install ufw
sudo apt-get install ntp
```

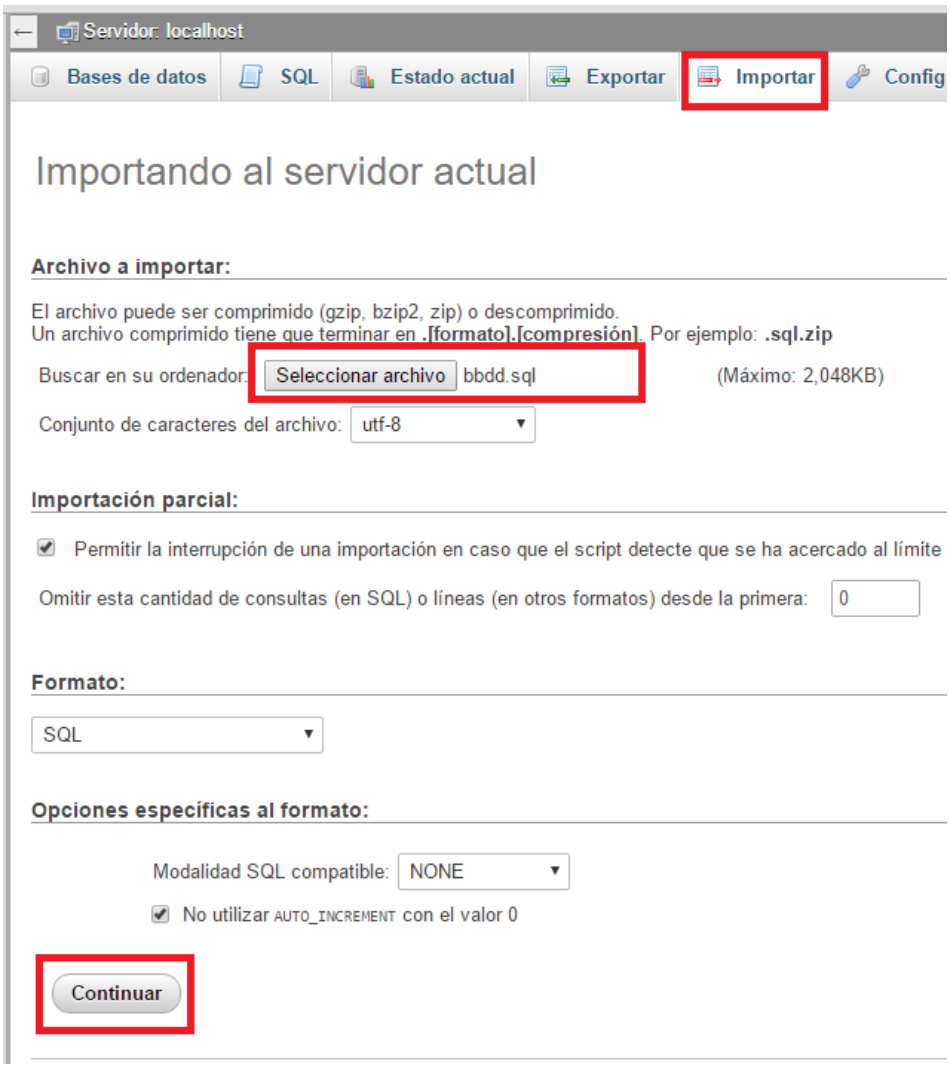
Para que resulte más cómodo, se puede hace la instalación a través de un único comando que englobe todos los paquetes.

```
sudo apt-get install -y apache2 mysql-server mysql-client php5 libapache2-mod-php5
php5-mysql php5-curl php-pear php5-dev php5-mcrypt php5-common php5-redis git-core
redis-server build-essential ufw ntp
```

Apéndice B: Despliegue de la aplicación

Una vez se tienen todas las dependencias instaladas, es necesario importar el fichero `bbdd.sql` a la base de datos diseñada a tal efecto, esta tarea se puede realizar mediante cualquier cliente de bases de datos como, por ejemplo, el cliente web phpmyadmin.

Nos iremos a la dirección IP de nuestra RBPi o por defecto la de localhost.
`http://localhost/phpmyadmin/index.php`



Una vez importada la base de datos o BC, se tiene que copiar la carpeta emoncms que se adjunta con el proyecto al directorio de apache considerado, en nuestro caso /var/www/, lo que se realiza mediante el siguiente comando.

```
cp -rf emoncms /var/www/
```

Hecho esto, es posible ir a `http://localhost/emoncms/` y comprobar que se muestra la pantalla principal de login.

En cualquier caso, para la comunicación de los sensores con el CMS, se requieres una serie de drivers que se ejecutan de la siguiente manera.

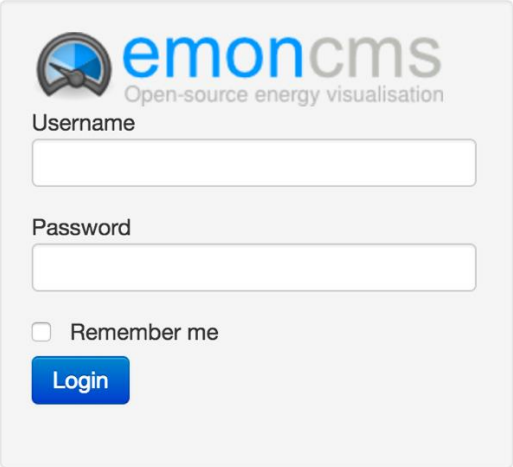
```
sudo python BMP180.py &
```

```
sudo python DHT22.py 22 17 &
```

Apéndice C: Uso de la aplicación

A continuación, se detalla el uso de la aplicación, haciendo especial énfasis en el módulo de IA. Para utilizar la aplicación, se dispone de un servidor donde se pueden realizar las diferentes pruebas y ver su uso.

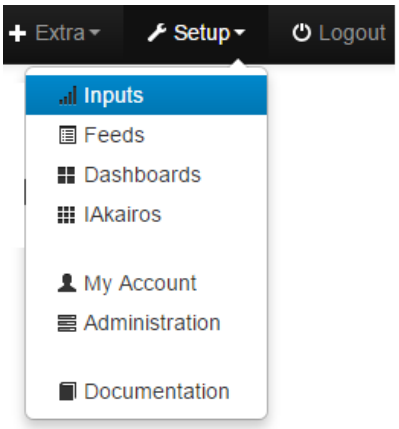
Lo primero que muestra la aplicación es el login, ya que es necesario identificarse,



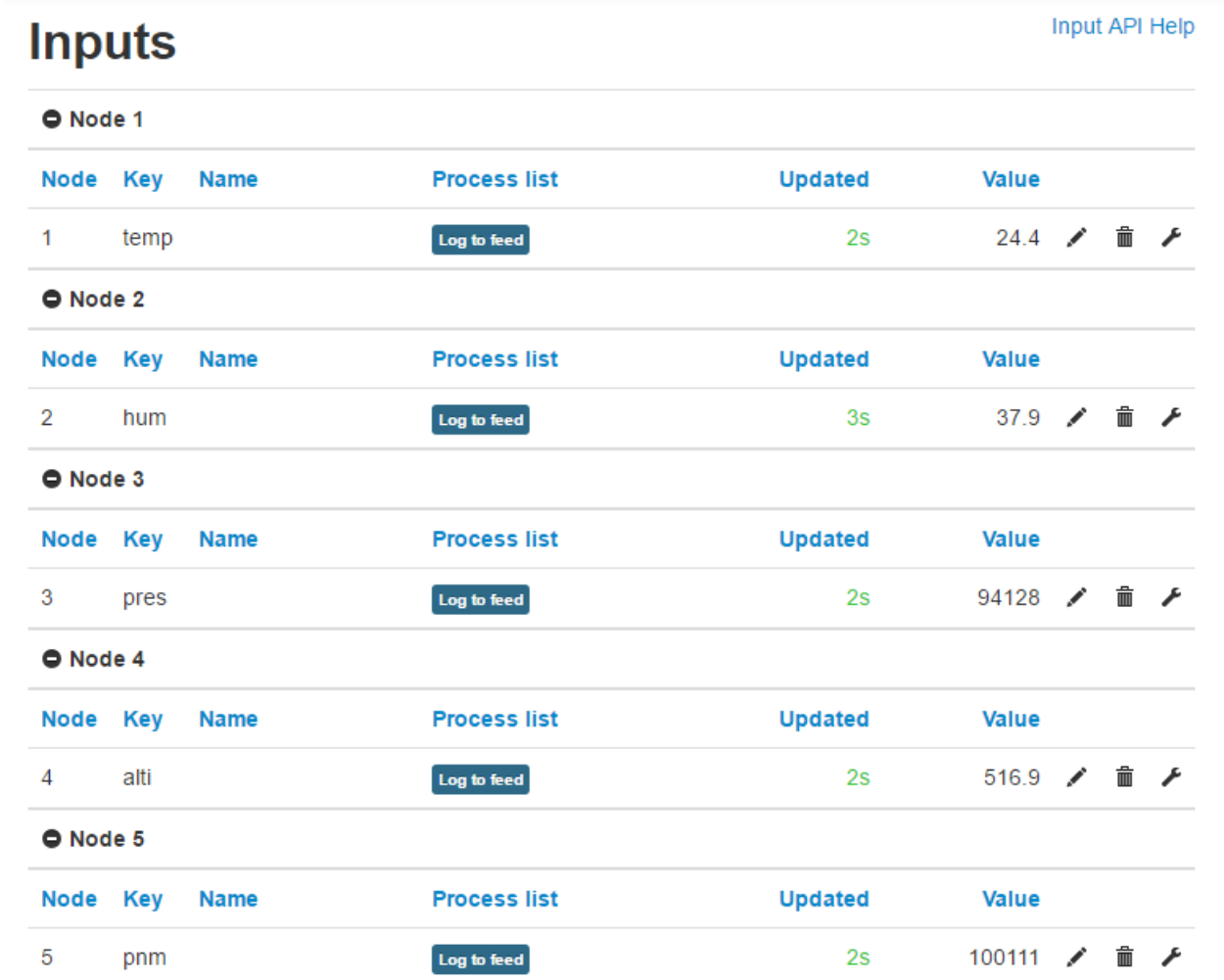
La identificación se puede realizar con el usuario emoncms, y la contraseña raspberry.

Una vez identificados, a continuación, se describe la forma de proceder para utilizar el módulo de IA.

Lo primero es comprobar la serie de datos de entrada que están llegando al CMS, ya que, si no existen datos de muestra para el entrenamiento, no es posible su utilización. Para ello nos vamos al menú “Setup” apartado Inputs.



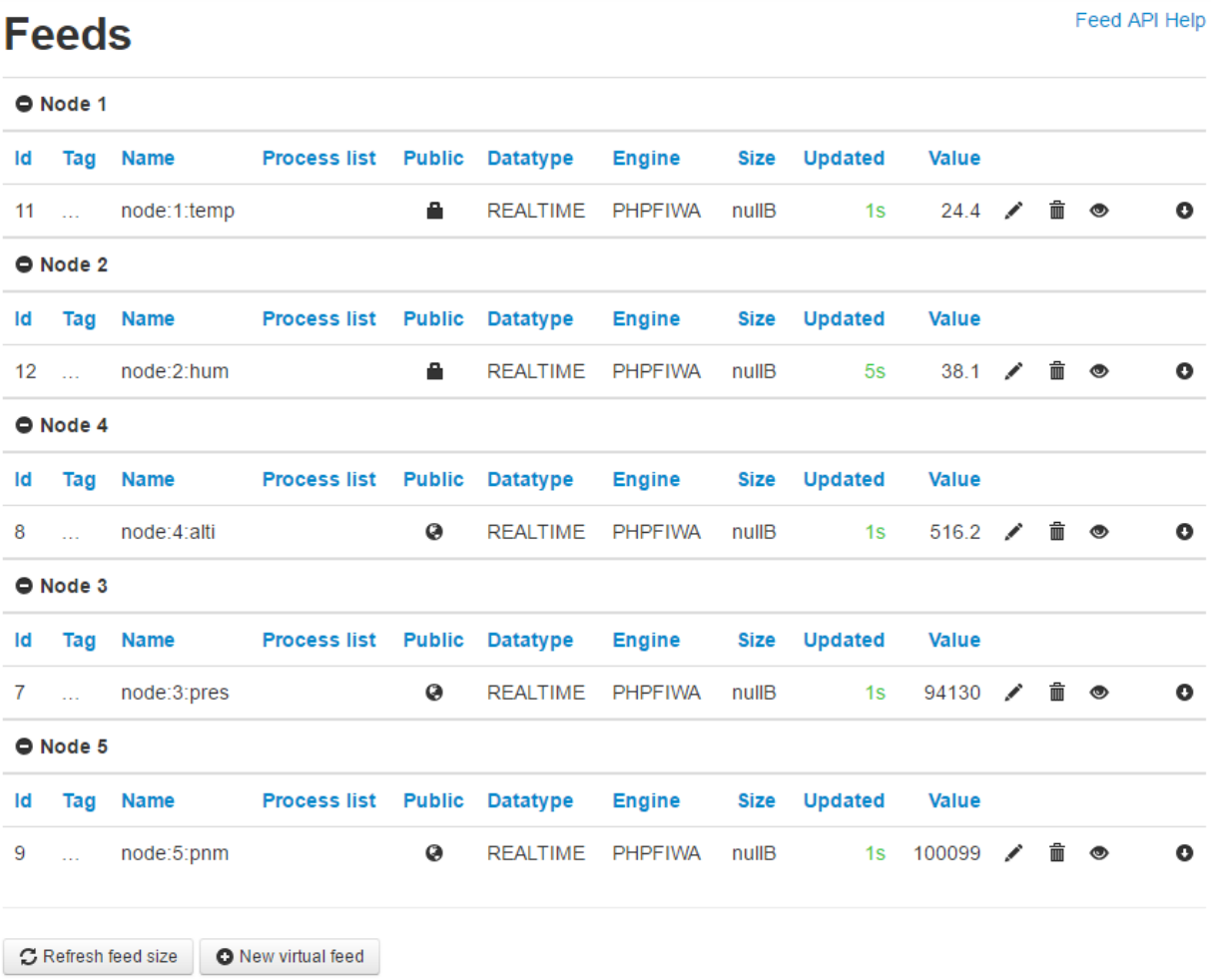
A continuación, aparecerá una lista de inputs como puede ser el ejemplo siguiente:



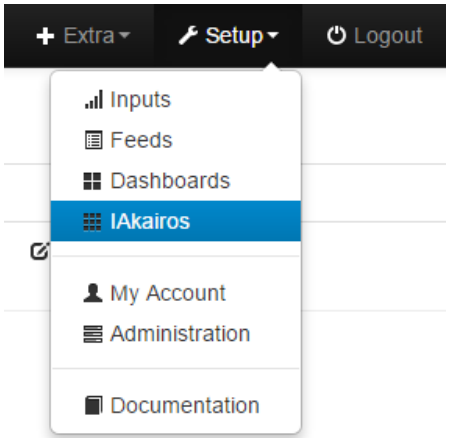
De esta forma es posible comprobar el correcto funcionamiento de los drivers, así como la llegada de la información proporcionada por los sensores remotos.

De estos inputs que llegan al sistema, tenemos que seleccionar los deseados e incluirlos como Feeds del sistema, ya que son estos feeds, los que la aplicación maneja y procesa.

Quedando los feeds de tal forma:

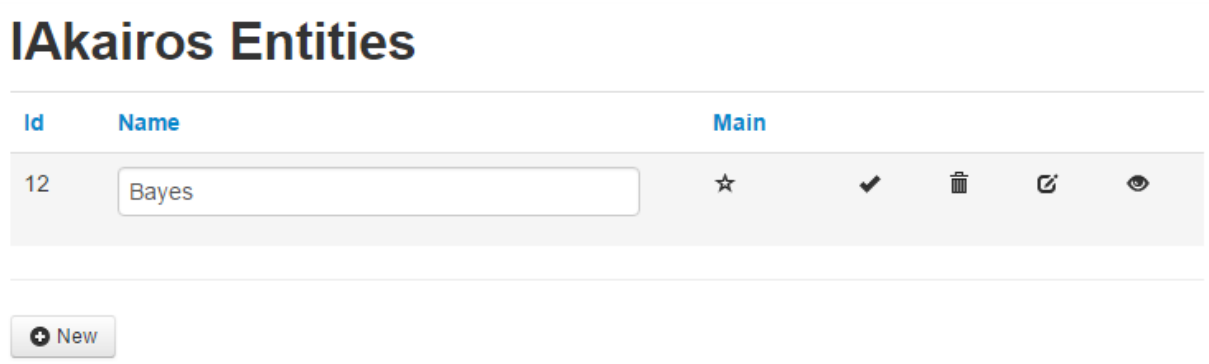


A continuación, se muestra el menú de “Setup”, seleccionando el nombre del módulo asociado a la presente aplicación, esto es IAkairos.

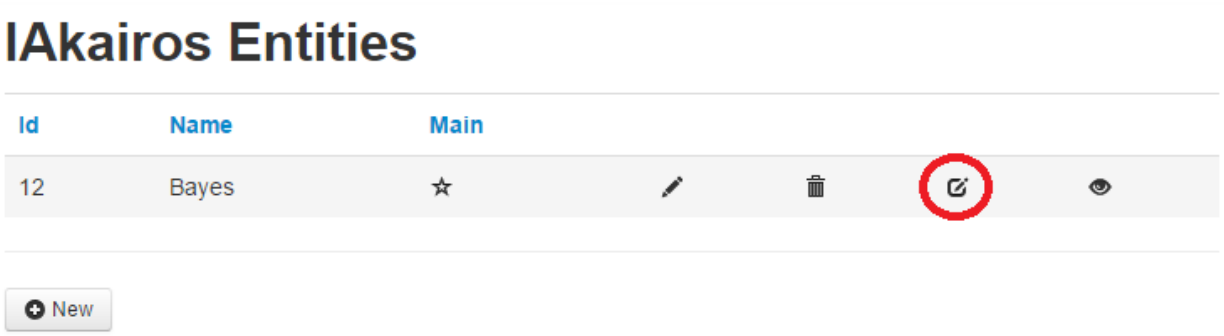


Inicialmente no existe ninguna entidad en este módulo, por tanto, se procede a crear una nueva. Para ello se pulsa el botón new, apareciendo una nueva Entity con nombre “no name”,

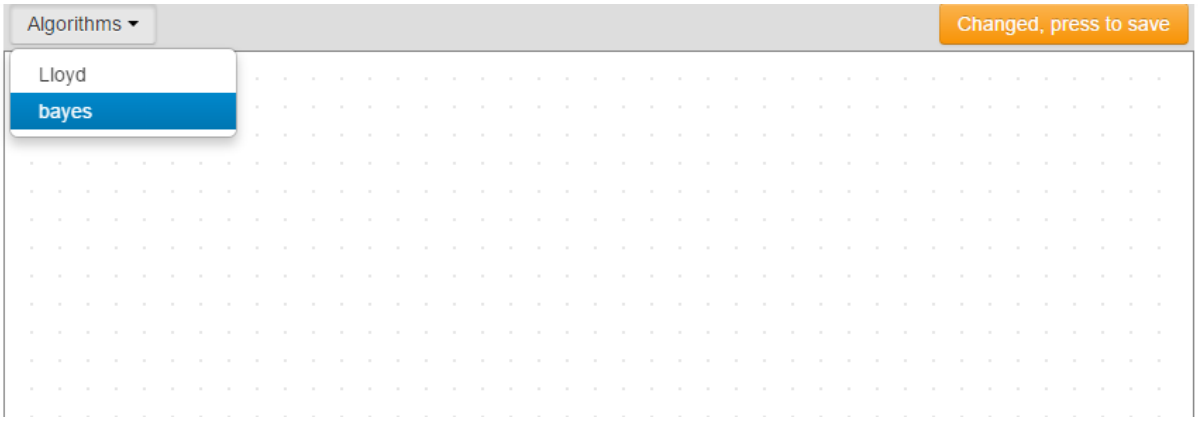
con posibilidades de modificarlo convenientemente. En este caso, se ha decidido que coincida con el nombre del algoritmo de IA a utilizar.



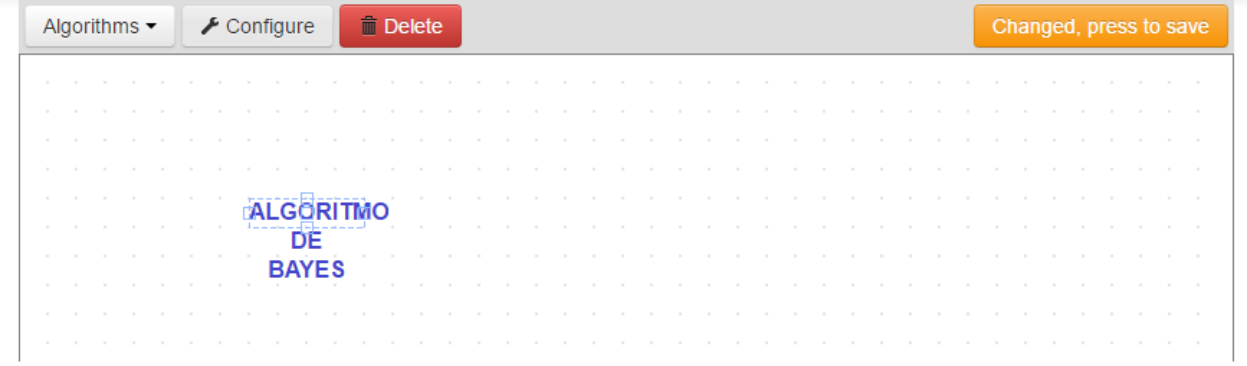
Una vez realizada la operación previa, se procede a su configuración, para ello, se pulsa el botón enmarcado con un círculo rojo, tal y como se muestra en la siguiente imagen.



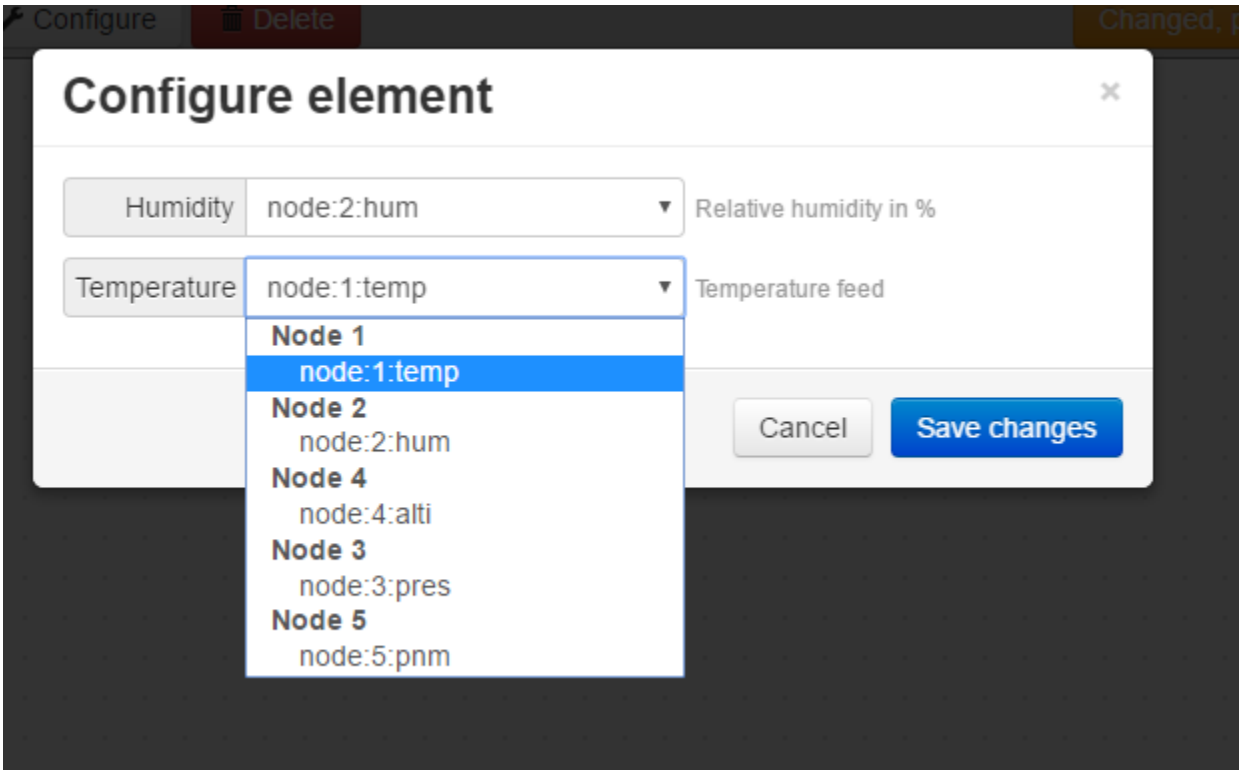
Llegados a este punto, se puede elegir qué tipo de algoritmo de los que existen implementados podemos usar, en nuestro caso se elige Bayes, ya que Lloyd no está implementado en esta versión, en este caso, sólo se muestra como ejemplo de otros posibles algoritmos.



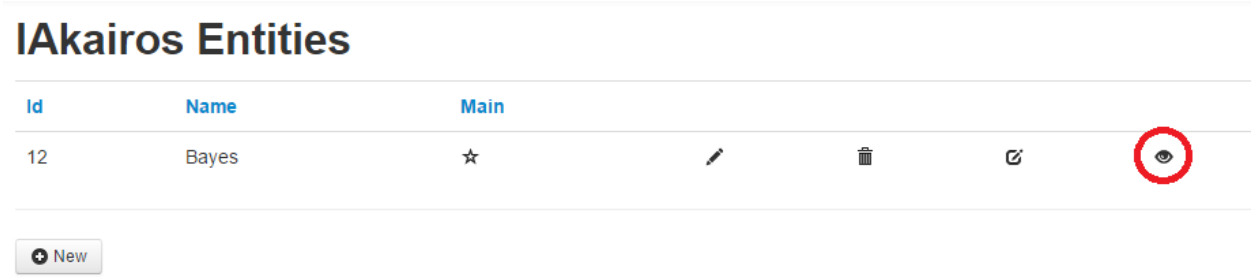
Se selecciona y se pincha sobre el rectángulo de puntos, para a continuación poder configurar qué inputs se van a utilizar para el algoritmo seleccionado. Para ello, se selecciona “Configure”.



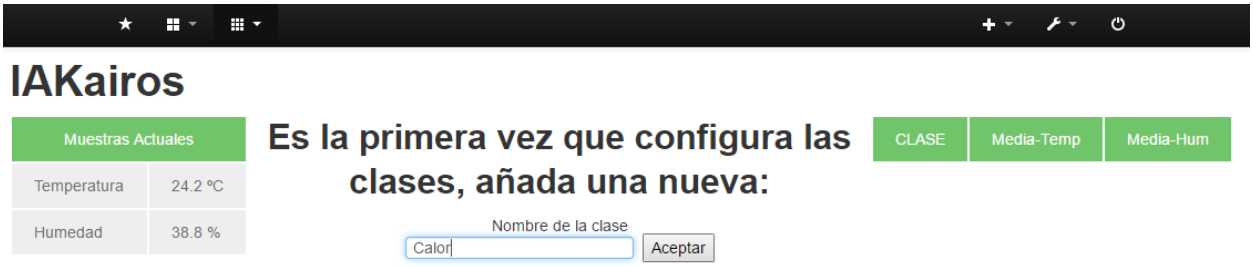
Bayes necesita humedad y temperatura, por lo tanto, se seleccionan las entradas que proporcionan esa información, una vez seleccionadas, se guardan los cambios realizados.



Seguidamente es necesario posicionarse en la “vista” desde el mismo panel anterior, señalado en la siguiente imagen con un círculo rojo.



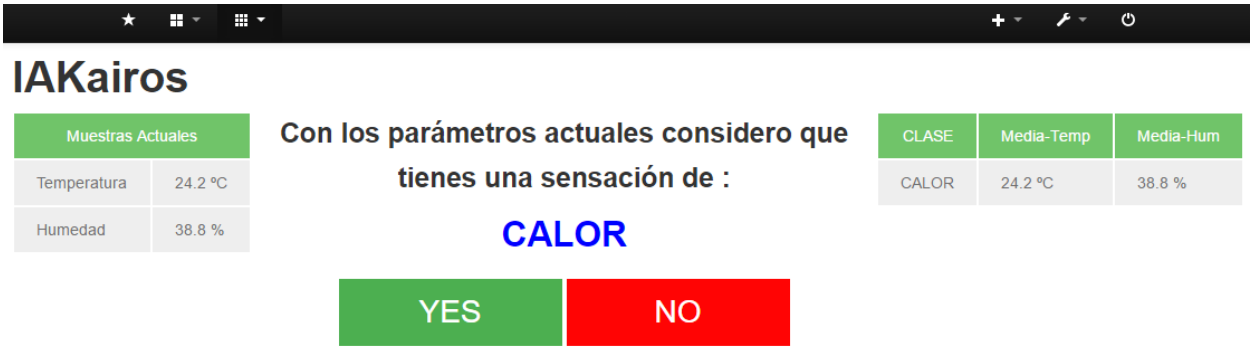
La primera vista disponible, es la que aparece a continuación,



En primer lugar, por un lado, se tiene la tabla de muestras actuales de temperatura y humedad. En el centro de la pantalla, es necesario definir una nueva clase, ya que no existe ninguna creada, mientras en la parte derecha, aparece una tabla con todas las clases que se van creando, así como los valores de sus respectivos centros, tal y como establece el algoritmo de Bayes.

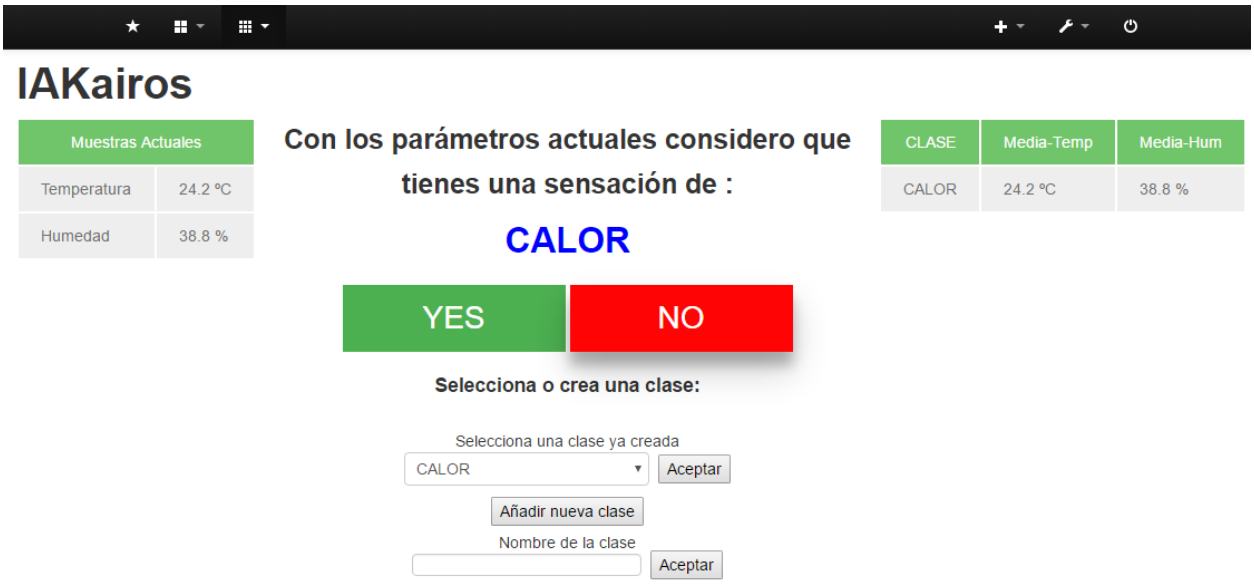
A continuación, se define una nueva clase, por ejemplo CALOR, de forma que el usuario en este momento ha decidido que con 24,2ºC de tempera y 36,8% de humedad, la sensación es de calor, que podría discrepar de otras percepciones personales.

Una vez aceptada, de forma automática se añade esa clase con esos datos, de forma que, en este caso, y con la llegada de datos similares, el sistema indica que la sensación es de calor (porque es la única clase existente en el sistema).



Como se puede aprecia, el sistema muestra la sensación correspondiente para el entrenamiento realizado, así como dos opciones para verificar si esa “predicción” es correcta, o no, en este último caso, para poder continuar con el proceso de aprendizaje.

Al seleccionar NO, aparece el siguiente panel, para poder cambiar de clase, o añadir una nueva.



Una vez mostradas las partes básicas de la vista, se procede a realizar el correspondiente proceso de entrenamiento, cuyo objetivo es conseguir un mayor ajuste en relación a cada sensación individual.

Seguidamente, se muestra una imagen con diferentes pruebas realizadas y cómo lo va tratando el sistema.

En este momento, tal y como puede observarse, se dispone ya de diferentes clases definidas (frio, templado, calor, mucho calor) con sus correspondientes centroides de clase, que son los parámetros aprendidos por el método de Bayes.

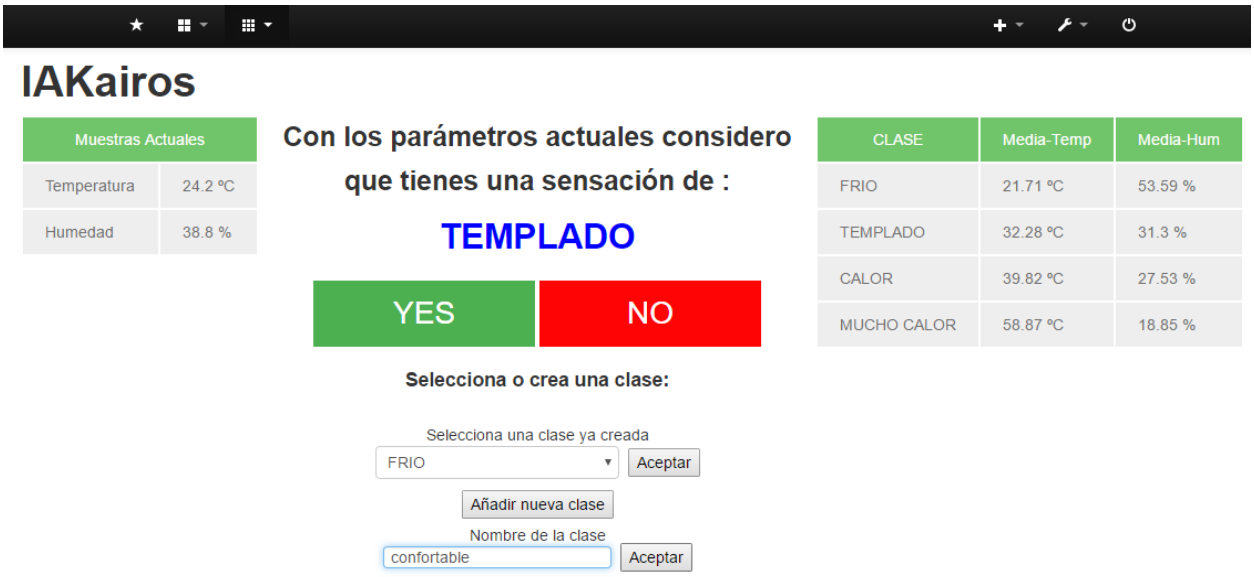
Por poner un ejemplo, con las muestras disponibles en el instante mostrado (24,2ºC de Temperatura y 38,6% de humedad) la sensación percibida de templado es correcta, procediendo en este caso a seleccionar el botón YES, para añadirlo al sistema.

Al seleccionar YES, aparece un mensaje informando de que los centros han sido actualizados correctamente.

Puede observarse en la tabla de las medias como automáticamente, se han reajustado los valores medios de los centros de la clase templado.

Si, por lo contrario, la sensación percibida no se corresponde con templado, sino con otra categoría, se puede pulsar el botón NO y seleccionar a qué otra clase de las ya definidas corresponde.

Si se diera el caso, de que ninguna de las categorías creadas anteriormente son satisfactorias, se selecciona, añadir nueva clase, para introducir esa nueva clase o categoría.



En este caso, se procede a crear la clase “confortable”. Una vez añadida, ésta se inserta en la tabla de clases con los parámetros de la muestra y cambia la “predicción” de templado a confortable.

Este cambio se produce porque en este caso resulta ser la clase más próxima a los datos del momento.

Tal y como se ha mencionado previamente, los resultados y por tanto las decisiones, dependen de sensaciones del usuario.

Apéndice D: Código relevante.

Código del driver del BMP180, muy similar al del DHT22 si obviamos la implementación de las respectivas lecturas de datos.

```
def main():

 temperatura = sensor.read_temperature()
 presion = sensor.read_pressure()
 altitud = sensor.read_altitude()
 presnm = sensor.read_sealevel_pressure(altitud)

 API_key = "9d990d9b350271774153b8ace5df7d7d";

 while(True):
   url1 = "http://localhost/emoncms/input/post.json?node=1&json={temp:" +
str(temperatura) + "}&apikey=" + API_key
   url2 = "http://localhost/emoncms/input/post.json?node=3&json={pres:" +
str(presion) + "}&apikey=" + API_key
   url3 = "http://localhost/emoncms/input/post.json?node=4&json={alti:" +
str(altitud) + "}&apikey=" + API_key
   url4 = "http://localhost/emoncms/input/post.json?node=5&json={pnm:" + str(presnm)
+ "}&apikey=" + API_key

   time.sleep(3)

   temperatura = sensor.read_temperature()
   presion = sensor.read_pressure()
   altitud = sensor.read_altitude()
   presnm = sensor.read_sealevel_pressure(altitud)

   urllib2.urlopen(url1)
   urllib2.urlopen(url2)
   urllib2.urlopen(url3)
   urllib2.urlopen(url4)

if __name__=="__main__":
  main()
```

6. Bibliografía

[1] "Internet of things," [Online]. Disponible en: https://en.wikipedia.org/wiki/Internet_of_Things (accedido Junio 2016).

[2] "Internet de las cosas," [Online]. Disponible en: https://es.wikipedia.org/wiki/Internet_de_las_cosas (accedido Junio 2016).

[3] "Sistema de gestión contenidos," [Online]. Disponible en: https://es.wikipedia.org/wiki/Sistema_de_gestión_de_contenidos (accedido Junio 2016).

[4] "Content Management System," [Online]. Disponible en: https://en.wikipedia.org/wiki/Content_management_system (accedido Junio 2016).

[5] "EmonCMS," [Online]. Disponible en: https://emoncms.org/ (accedido Junio 2016).

[6] "Inteligencia Artificial," [Online]. Disponible en: https://es.wikipedia.org/wiki/Inteligencia_artificial (accedido Junio 2016).

[7] G. Pajares and M. Santos, Inteligencia artificial e ingeniería del conocimiento, Madrid: Ra-Ma, 2005. (accedido Junio 2016).

[8] "Raspberry Pi," [Online]. Disponible en: https://www.raspberrypi.org/ (accedido Junio 2016).

[9] "GPIO," [Online]. Disponible en: https://www.raspberrypi.org/documentation/hardware/raspberrypi/gpio/README.md (accedido Junio 2016).

[10] "GPIO," [Online]. Disponible en: https://es.wikipedia.org/wiki/GPIO (accedido Junio 2016).

[11] "I2C," [Online]. Disponible en: https://es.wikipedia.org/wiki/I%C2%B2C (accedido Junio 2016).

[12] Bosch Sensortec, "BMP180 Digital pressure sensor," [Online]. Disponible en: https://cdn-shop.adafruit.com/datasheets/BST-BMP180-DS000-09.pdf (accedido Junio 2016).

[13] L. Aosong Electronics Co., "DHT22," [Online]. Disponible en: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf (accedido Junio 2016).

[14] "Placa de pruebas," [Online]. Disponible en: https://es.wikipedia.org/wiki/Placa_de_pruebas (accedido Junio 2016).

[15] "RPi-Monitor," [Online]. Disponible en: http://rpi-experiences.blogspot.com.es/p/rpi-monitor.html (accedido Junio 2016).

[16] "OpenEnergyMonitor," [Online]. Disponible en: https://openenergymonitor.org/emon/ (accedido Junio 2016).

[17] "Comunicación bus I2C," [Online]. Disponible en: http://robots-argentina.com.ar/Comunicacion_busI2C.htm (accedido Junio 2016).

[18] "Python Software Fundation," [Online]. Disponible en: https://www.python.org/ (accedido Junio 2016).

[19] "Adafruit," [Online]. Disponible en: https://www.adafruit.com/ (accedido Junio 2016).

[20] "Adafruit BMP Driver," [Online]. Disponible en: https://github.com/adafruit/Adafruit_Python_BMP (accedido Junio 2016).

[21] "Adafruit DHT Driver," [Online]. Disponible en: https://github.com/adafruit/Adafruit_Python_DHT (accedido Junio 2016).

[22] "Adafruit GPIO Library," [Online]. Disponible en: https://github.com/adafruit/Adafruit_Python_GPIO (accedido Junio 2016).

[23] R. S. Pressman, Ingeniería del software : un enfoque práctico, McGraw-Hill, 2010 (accedido Junio 2016).

[24] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach (accedido Junio 2016).

[25] G. Pajares and J. Cruz, Aprendizaje Automático: un enfoque práctico, RA-MA, 2010 (accedido Junio 2016).

[26] R. O. Duda, P. E. Hart and D. G. Stork, Pattern Classification, John Wiley & Sons, 2001 (accedido Junio 2016).

[27] "Teorema de Bayes," [Online]. Disponible en: https://es.wikipedia.org/wiki/Teorema_de_Bayes (accedido Junio 2016).

[28] G. Pajares and J. M. de la Cruz, Ejercicios resueltos de visión por computador, Madrid: Ra-Ma, 2007 (accedido Junio 2016).

[29] G. Pajares and J. M. de la Cruz, Visión por computador: Imágenes digitales y aplicaciones, Madrid: Ra-Ma, 2007 (accedido Junio 2016).

[30] "OpenStreetMap," [Online]. Disponible en: https://www.openstreetmap.org/ (accedido Junio 2016).